Introduction to Intelligent Robotics

Herman Bruyninckx and Joris De Schutter

Katholieke Universiteit Leuven
Departement Werktuigkunde

Notes for the "Robotics" course (HB46)
`http://www.mech.kuleuven.ac.be/~bruyninc/hb46/`

The dream of researchers in robotics is to create the ultimate human-like robot. Irrespective of whether they admit it or not, and irrespective of whatever moral implications and complications this might bring. But... this ultimate robot will not be there tomorrow! However, all of its basic building blocks are there already: mechanical skeletons, motors and sensors, computer brains, reasoning and control algorithms, ... All of them have reached a state from which researchers all over the world can start constructing the "DNA" of the humanoid robot. All we need is time and a lot of effort to make it grow and reach its full potential. This course gives students the opportunity to discover this robot DNA, and meet the embryos it has already created. But be warned: many regions in the robot's double helix are still under construction and the road is heavy.

Have a nice trip!

7th edition—1 October 2001

Contents

Chapter 1

Introduction

A large part of your life, you are moving around and manipulating objects: preparing a meal, typing a text, washing the dishes, building a house, assembling a bicycle, etc. Take a closer look at these actions, and you'll notice that they require a huge number of skills: recognizing the objects; estimating where they are; discriminating between "obstacles" and the objects you are really interested in; planning what to do and how to do it; moving arms, hands, and fingers to the right places and in the correct way in order to bring the objects where you want them; in the meantime continuously adapting the motions to inputs from all of your senses (vision, touch, and hearing being the most relevant for manipulation tasks). You seldom pause to think about the utmost complexity of these "automatically" executed actions. Indeed, none of these actions is easy to automate. This text will teach why. It covers the fundamentals of how people have tried to reach this ultimate goal of the " artificially intelligent machine": the *robot.* The text attacks the problem from both the *engineering* side of the problem (i.e., those things for which engineers have come up with solutions that basically rely on the application of the laws of physics or the theorems of mathematics) and the *artificial intelligence* side (where human behaviour has been the major source of inspiration). The real breakthrough of "intelligent" robots will most probably only be achieved if the engineering and AI sides can meet and join their forces. Hence, the major goal of this text is to fill the gap between both approaches. Students with an engineering background will learn what "information" means, and "uncertainty," and "reasoning;" other students will learn the principles of how machines move and how sensors make models of the real world.

1.1 When was the robot born?

For economic or scientific purposes—or just for fun—mankind has, for centuries already, tried to make machines autonomously perform manipulative actions not unlike the ones mentioned in the previous paragraph, and to mimic the human behaviour as closely as possible. However, only since the advent of the hydraulic and electric motors, and especially of the electronic computer, this automation of human abilities has become realistically feasible. Hence, a whole new science of making "mechanical slaves" has come into existence. Because man's imagination is always decades ahead of his capabilities, this science got its name long before it really became a scientific discipline worth the name: the *robot* appears on stage (literally) for the first time (1921) in the theatre play *R.U.R.* (Rosum's Universal Robots) by the Czech writer Karel Čapek, [2]. The Czech word *robota* means "regular forced labor of serfs—poor peasants on the land of their lords in a feudal society."[1] The word "Rosum" does not only represent the name of a person, but it also means "reason" in Czech as well. So, already in its

[1]Source: Czech Society For Cybernetics and Informatics, July 1994.

first confrontation with the world, the robot shows both sides of machine intelligence: cheap and versatile labour, but also potential misuse and ethical problems. It's interesting to note that R.U.R.'s robots are the product of genetic engineering rather than mechatronic engineering: the robots are created from human genes from which all emotional factors have been removed (or rather, their creators were convinced they removed them all...).

Outside of the theatre, real industrial robots were developed in the fifties and sixties; the designs that populate most of the automobile assembly plants were invented in the late sixties. The major developments took place in the American car industry (especially *General Motors*), and in the nuclear industry. The real industrial robot looks quite different from the robot encountered in science fiction films—quite a disappointment in fact. However, recently there is a growing interest for developing *humanoid*[2] robots, especially in Japan. For example, after ten years of research, *Honda Corporation* in 1997 presented its first two-legged ("bi-ped") walking robot, which looks very much like a human in a space suit.[3] This robot comes closer to our intuitive notion of a robot, and that's exactly the purpose of this (almost exclusively Japanese) research: make the machine look and act like a human, in order to decrease the threshold for humans to use robots. However, the capabilities of this humanoid robot are still far below those of a real human: it can walk and that's just it... Nevertheless, it should come as no surprise that your kids will want to buy "intelligent" toy robots a couple of years from now.[4]

1.2 What is robotics?

The word *robotics* was coined by the Russian-born scientist and science fiction writer Isaac Asimov (1920–1992), in the early 1940s; again, much before any real robots existed. As a scientific research area robotics is currently an extremely *interdisciplinary* activity. It is a rich mixture of exact and applied sciences such as geometry, mechanical design, kinematics, statics and dynamics, (numerical) linear algebra, system identification and control theory, real-time operating systems, electronics, software engineering and Artificial Intelligence, deterministic and stochastic filtering or sensor signal processing, etc. Roughly speaking, robotics has two faces: (i) the mathematical and engineering face, which is quite "standardized" in the sense that a large consensus exists about the tools and theories to use, and (ii) the AI face, which is rather poorly standardized, not because of a lack of interest or research efforts, but because of the inherent complexity of "intelligent behaviour." Research in engineering robotics follows the *bottom-up* approach: existing and working systems are extended and made more versatile. Research in artificial intelligence robotics is *top-down*: assuming that a set of low-level primitives is available, how could one apply them in order to increase the "intelligence" of a system. The border between both approaches shifts continuously, as more and more "intelligence" is cast into algorithmic, system-theoretic form. For example, the response of a robot to sensor input was considered "intelligent behaviour" in the late seventies and even early eighties. Hence, it belonged to A.I. Later it was shown that some simple sensor-based tasks such as surface following or visual tracking could be formulated as control problems with algorithmic solutions. From then on, they did not belong to A.I. any more.

Robotics is roughly based on the integration of knowledge from the domains of *sensing*, *planning*, and *control*. Moreover, in classical, engineering-oriented robotics, *modelling* is the central tool for this integration: the theoretical descriptions are in terms of a *physical model*—i.e., an abstraction of the real world— containing only those properties of the real world that are relevant for a given application. However, one usually does not work on the physical models directly, but rather on yet another level of abstraction: the *coordinate representations* of this physical model, sometimes called the "mathematical model". Moreover, often only *linear* models are used. Linear models enormously simplify proofs and algorithms. But complex, life-like behaviour is often generated by the nonlinear apects of the system only!

[2]`http://www.androidworld.com/prod01.htm`
[3]`http://www.honda.co.jp/english/technology/robot/index.html`
[4]`http://www.aibo-europe.com/`

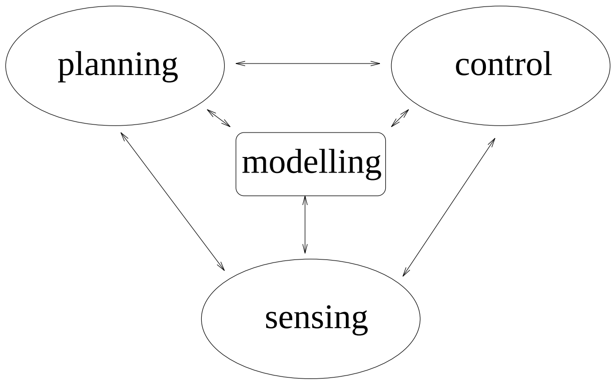


Figure 1.1: Planning, sensing and control are the major components of any model-based robotic system.

A relatively new branch in robotics investigates the applicability of *model-free* approaches (which are inherently nonlinear): learning, genetic algorithms, neural networks, etc. These approaches are not discussed in this introductory text.

Let's give an example of the chain "*real world–physical model–coordinate model*." A real robot *could* physically be modelled as:

1. A set of rigid links, connected by perfect, frictionless joints that allow the device to move.
2. Each link has a certain mass and inertia.
3. The joints are driven by electrical motors with known (or rather, estimated) characteristics.
4. The position of the joints is sensed by encoders or resolvers, and its velocity by tachometers.
5. A computer digitizes and processes the measurements, uses them to generate control signals that keep the robot on its desrired path, and interacts with the human operator.

The corresponding coordinate descriptions *could* be:

1. A geometrical model of directed lines ("vectors") representing the links and connecting reference frames at the positions of the joints; the relative orientation of the lines and frames depend on the robot joint angles.
2. A set of "centre of mass" reference frames, to which mass and inertia matrices are assigned.
3. A set of RCL loops in the motors (resistors, capacitances, inductances), with ideal current and/or voltage sources.
4. Discrete or continuous stochastic variables, with given "noise" characteristics that represent the uncertainty about the world they model.
5. A set of PID control algorithms.

The above-mentioned physical and coordinate models are not unique: in general, multiple physical models exist for the same real world system, and, in turn, multiple coordinate representations exist for each physical model. *No* physical or coordinate model is ever completely correct!

1.3 Where is the intelligent robot?

Although the motion capabilities of current robotic devices are orders of magnitude better than those of a human being—i.e., they can be made faster, more accurate, and stronger than any human—their "intelligence" and "dexterity" are infantile, to say the least. The gap between the commercial and the academic state of the art in robotics is an order of magnitude smaller than the gap with the science fiction robots, but still considerably large: approximately ten years. And this gap tends to grow even larger all the time. The major cause is the difference in *reliability* between the simple (and hence robust) position-controlled robot arms employed in industry on the one hand, and the complex sensor-guided (and hence error-prone) research prototypes on the other hand: no advanced sensor processing and interpretation techniques currently exist that meet the industrial norm of "6σ" reliability—i.e., a Mean Time Between Failure (MTBF) of over 20 000 hours.

The general public's expectations towards robotic devices have been enormously inflated by the fantasies and special effects of the entertainment industry. The human-like machines—in shape as well as in behaviour—brought on stage in these science fiction stories are indeed nothing more than what their name suggests: fiction. The realisation of intelligent machines will require many more generations of highly educated and qualified scientists and engineers, able to understand and integrate the progress in all those diverse sciences and technologies that, together, make up robotics. To prepare the students for this challenge is exactly the goal of this text.

Despite the somewhat pessimistic remarks above, there does exist one particular theoretical toolhat has succeeded the last couple of years to merge the "low-level" control work with the "high-level" Artificial Intelligence work. This tool is *Bayesian probability theory.* The good news is that is can be used equally well to cope with the uncertainty inherent in sensor-based control, as with the uncertainty inherent in reasoning about actions. The bad news is that (i) due to its mathematical rigour, it can be quite difficult to apply to a given robotics problen, and (ii) the majority of robotics researchers are not familiar with the Bayesian approach.

1.4 What kind of robots?

This text basically investigates the following types of robotic devices: serial robot arms, parallel robot arms, humanoid robots, and mobile robots. Except for the humanoid robot, these are the only types that are currently and routinely used in industrial or service applications.

1.4.1 Serial manipulators

Serial manipulators are by far the most common industrial robots. Often they have an anthropomorphic mechanical arm structure (Fig. 1.2), i.e., a serial chain of rigid links, connected by (mostly revolute) joints, forming a "shoulder," an "elbow," and a "wrist." Their main advantage is their *large workspace* with respect to their own volume and occupied floor space. Their main disadvantages are (i) the *low stiffness* inherent to an open kinematic structure, (ii) *errors* are accumulated and *amplified* from link to link, (iii) the fact that they have to carry and move the *large weight* of most of the actuators, and (iv) hence, the relatively *low effective load* that they can manipulate.

It requires at least *six* degrees of freedom to place the manipulated object in an arbitrary position and orientation in space—or rather, in the workspace of the robot. Hence, many serial robots have six joints. However, the most popular application for serial robots in today's industry is *pick-and-place* assembly: the robot takes an object from a component feeder system (such as a conveyor belt or a vibrating bowl feeder) and brings it to its final place in an assembly setup. This requires only *four* degrees of freedom maniplators: a vertical motion (translation along Z) and the three planar degrees of freedom (translation along X and Y, and rotation about Z). For this purpose, special assembly robots are built, of the so-called *SCARA* type (Selective Compliance Assembly

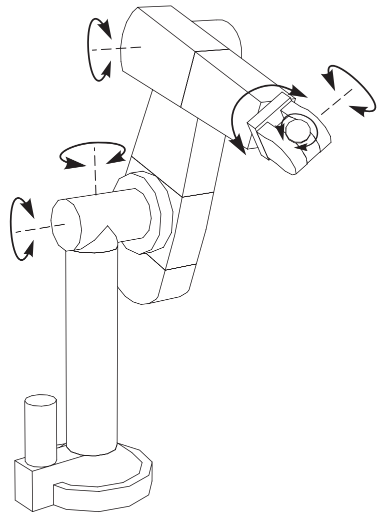


Figure 1.2: Serial robot.

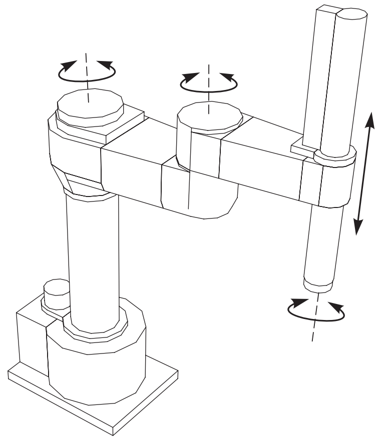


Figure 1.3: SCARA robot.

Robot Arm), (Fig. 1.3). This design was invented by Prof. Hiroshi Makino of Yamanashi University, Japan, in 1972.

1.4.2 Parallel manipulators

A parallel manipulator consists of a fixed "base" platform, connected to an "end effector" platform by means of a number of "legs" (Fig. 1.4). These legs often consist of an actuated prismatic joint, connected to the platforms through passive (i.e., not actuated) spherical and/or universal joints (Fig. 1.5). Hence, the links feel only traction or compression, not bending, which increases their position accuracy and allows a lighter construction. The actuators for the prismatic joints can be placed in the motionless base platform, so that their mass does not have to be moved, which again makes the construction lighter. Parallel manipulators have (in principle!) high structural stiffness, since the end effector is supported in several places at the same time. All these features result in manipulators with a *high-bandwidth* motion capability. Their major drawback is their *limited workspace*, because the legs can collide (with each other and with themselves) and, in addition, each leg has five *passive joints* that each have their own mechanical limits. Major industrial applications of these devices are airplane and automobile simulators. They also become more popular in (i) high-speed, high-accuracy positioning with limited workspaces, such as in assembly of Printed Circuit Boards, (ii) as *micro manipulators* mounted on the end effector of larger but slower serial manipulators, and (iii) as high-speed/high-precision milling machines.

The example above is called a *fully parallel* manipulator, because each leg has only one actuated joint. Alternatively, each of the legs could be a serial robot with more than one actuated joint. Fully parallel manipulators with two-by-two intersecting prismatic legs, (Fig. 1.4), are often called *Stewart platforms*, referring to the author of the "first" article describing the application of a parallel manipulator as a flight simulator, [7]. The same fully parallel design was already used some years earlier in a tire testing machine, as reported by Gough [4]. Hence, some authors prefer the name *Stewart-Gough* platform to honour both.

In contrast with serial manipulators, a (fully) parallel manipulator needs *at least* six legs (with six degrees of freedom). Otherwise, the structure is not stable: it can move without changing the lengths of the legs.

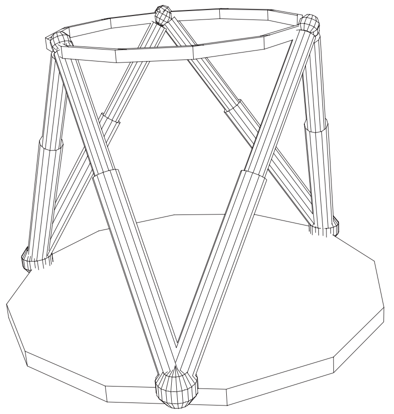


Figure 1.4: Parallel robot of the *Stewart-Gough* type.

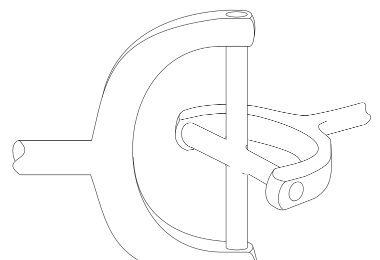


Figure 1.5: Universal ("Hooke") joint.

1.4.3 Mobile robots

Roughly speaking, mobile robots are automated cars or bicycles, i.e., wheeled vehicles with two degrees of freedom: (i) forward-backward driving and rotation of the steering wheel (e.g., automatic cars), or (ii) two independently actuated wheels (e.g., electric wheelchairs). Mobile robots are currently mainly used to transport materials over large factory floors. In this case, a planar model suffices to represent a mobile robot's motion freedom. Outdoor navigation on rough terrain (e.g., planetary rovers) requires *three-dimensional* motion (and sensing!) capabilities.

Mobile robots are prime examples of *nonholonomic* systems: they can not move sideways (i.e., they have only a *two-dimensional* space of instantaneous motion freedom), but yet they can attain every possible position and orientation in their plane of motion (i.e., they can attain all configurations in a *three-dimensional* space). There exist so-called *omni-directional* mobile robots, with special wheel designs that do allow instantaneous motion in three directions, e.g., [1, 6, 8, 9]. These devices have not been very successful until now, because of many practical problems: clearance with the floor, difficult to add pneumatic tires, poor power transmission, etc. But the novel *powered caster* design, [5], seems to be able to solve all the practical problems.

Another feature of a nonholonomic system is that it can generate a net motion (i.e., a change in its "external" parameters, being its position and orientation) by a *cyclic* motion of its "internal" parameters (i.e., its wheel angles). Other nonholonomic systems are: a spinning satellite, a falling cat, a cutting knife, a diver or gymnast performing somersaults or spins. Nonholonomic systems require advanced planning and control approaches, many of which rely on nonlinear geometry.

1.4.4 Humanoid robots

Roughly speaking, a robot is called *humanoid* if it looks a lot like a human. Since Karel Čapek introduced the word "robot" to denote slaves derived from real human beings, that word has commonly been used to indicate human-like machines, with high intelligence and dexterity. Two decades before Čapek already, Otto Fisher described the dynamics of human motion, [3]. However, robotics as an engineering discipline really started only after the second World War, and has long been mostly limited to the simple robot arms used in the nuclear and manufacturing industries. Only the last decade, the state-of-the-art in engineering has become sufficiently mature and miniaturized in order to make *humanoid* machines possible. A robot is called "humanoid" if it has two legs, two arms, and a head, and uses them in a way similar to human beings: the legs to walk, the arms to manipulate objects, and the head to see and hear.

Currently, the resemblance is at best formal and superficial: no machine approaches the intelligence of a human. However, the psychological effect of humanoid robots is large: humans seem to have much less problems to interact with a human-like robot than with a more classical robot arm, even if the functionalities of both are similar.

From the mechanical point of view, the humanoid robot is different from the serial and parallel architectures in that its topology is a *tree* (Fig. 1.6), i.e., there is only one path from any one link to any other link. A tree structure is the simplest extension of the serial structure, and requires only slightly more complex algorithms for its modelling and control.

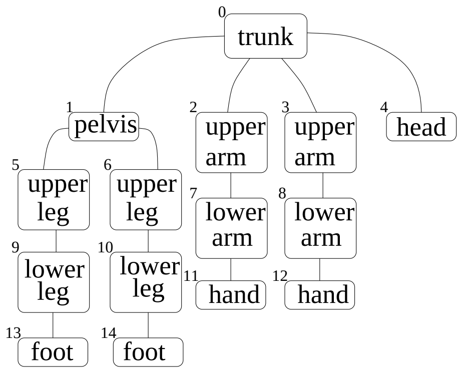


Figure 1.6: Tree-structure topology of a humanoid robot.

1.5 Conventions and notations

The following paragraphs explain the conventions and notations that will be used in the rest of the text.

References. Each chapter ends with a section containing all references used in that chapter. References are given between square brackets, like this: [2].

Vectors, matrices A boldface lowercase letter (e.g., "$\boldsymbol{a}$") denotes a *column of coordinates*. Several distinct types of coordinates are used:

1. The *coordinates of a point*, i.e., an ordered column of three numbers representing the directed line element from the origin of a reference frame to the position of a point. Both the point itself as well as its coordinates will often be denoted by the same notation, $\boldsymbol{p}$.

2. The *homogeneous coordinate vector*, i.e., the ordered *column* of *four* numbers, the last of which is always "1", $\boldsymbol{p} = (\boldsymbol{p}_x \ \boldsymbol{p}_y \ \boldsymbol{p}_z \ 1)^T$, and which represent the same coordinates as above. Homogeneous coordinates have some algebraic advantages, as will become clear in later Chapters.

3. The *coordinates of a vector*, i.e., an ordered column of three numbers, $\boldsymbol{v} = (\boldsymbol{v}_x \ \boldsymbol{v}_y \ \boldsymbol{v}_z)^T$, representing the vector's coordinates in a given reference frame.

4. The *coordinates of a line vector*, i.e., a vector with a direction and a magnitude, and bound to a fixed line in space.

5. The *coordinates of a free vector*, i.e., a vector with a direction and a magnitude, but whose exact point of application has no physical relevance.

The difference between a "point" and a "vector" might seem strange at first sight. The distinction is as follows: the spaces of points (such as the plane of the floor, or the three-dimensional space we live in) have no particular point that serves as "origin" in natural way; vector spaces (such as the space of all velocities of a moving point) do have such a naturally defined origin (e.g., the zero velocity vector). Moreover, the "addition" of two points is not well defined (it depends on the *arbitrary* choice of origin, while adding two velocity vectors is well-defined, independently of the chosen (inertial!) origin. (This is a first example of the difference between coordinates properties on the one hand, and the structure of the physical model the coordinates represent, on the other hand!)

Unit vectors are vectors with unit magnitude, and are often denoted by $\boldsymbol{e}$. Since robots live in the three-dimensional space that surrounds us, this text uses mostly *three-vectors* (or four-vectors for homogeneous coordinates). However, rigid bodies have *six* degrees of motion freedom, and hence *six-vectors* occur frequently too. Six-vectors are written in an upright font—such as for example $\mathbf{t}$ or $\mathbf{w}$—and vectors of other or non-specified dimensions in a slanted font—such as $\boldsymbol{t}$ or $\boldsymbol{w}$.

An uppercase letter denotes a transformation within one single space, or between two different spaces. For example, $A(p, q) : \mathbb{R} \times \mathbb{N} \rightarrow \mathbb{R}$ is a (possibly nonlinear) function mapping a real number and a natural number to a real number. The same letter, but in a boldface font, denotes the matrix of A's coordinates if A is a *linear* transformation:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} & \dots & \boldsymbol{A}_{1n} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} & \dots & \boldsymbol{A}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{m1} & \boldsymbol{A}_{m2} & \dots & \boldsymbol{A}_{mn} \end{pmatrix}. \tag{1.1}$$

Two special examples are the n-dimensional identity and zero matrices:

$$\boldsymbol{1}_n = \left.\underbrace{\begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}}_{n}\right\} n, \qquad \boldsymbol{0}_n = \left.\underbrace{\begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}}_{n}\right\} n. \tag{1.2}$$

The index "n" is often omitted if the dimension is clear from the context. One single column of a matrix is a coordinate vector; it is usually denoted by the lowercase boldface letter corresponding to the letter denoting the matrix, or by the uppercase boldface letter of the matrix, together with a subscript indicating the column:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{a}_1 & \dots & \boldsymbol{a}_n \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_1 & \dots & \boldsymbol{A}_n \end{pmatrix}. \tag{1.3}$$

Frames $\{f\}$ is the shorthand notation for a frame called "f" in the three-dimensional space. The letters X, Y, and Z denote the three coordinate axes of a frame, and O denotes the origin of the frame. Hence, the extended notation for the frame "f" is: $\{O^f; X^f, Y^f, Z^f\}$. $\boldsymbol{e}_x, \boldsymbol{e}_y$ and $\boldsymbol{e}_z$ denote unit vectors along the positive X, Y and Z axes of a frame; if the name of the frame (e.g., $\{a\}$) is important, the notations $\boldsymbol{x}^a, \boldsymbol{y}^a$ and $\boldsymbol{z}^a$ are often used instead. Rigid bodies are sometimes denoted by the name of a reference frame that is rigidly fixed to the body.

Subscripts and superscripts A *leading superscript* to a transformation, a vector, or a matrix identifies the rigid body whose physical characteristics are described: ${}^a\boldsymbol{T}$. A *leading subscript* denotes the frame with respect to which a physical characteristic is expressed: ${}_f\boldsymbol{p}$. A *trailing superscript* indicates a feature of a frame, vector, etc.—e.g., $\{f\}^a$ says that the frame $\{f\}$ is considered fixed to rigid body a—or it enumerates the members in a set—e.g., $\boldsymbol{p}^1, \ldots, \boldsymbol{p}^n$ are the position vectors of a given set of n points. A *trailing subscript* denotes a coordinate component: $\boldsymbol{p}_x$ is the projection on the X axis of a position vector $\boldsymbol{p}$. So, the coordinates, with respect to the reference frame $\{a\}$, of the ith vector $\boldsymbol{p}^i$ in a set are given by the column vector

$$\begin{pmatrix} {}_a\boldsymbol{p}^i_x \\ {}_a\boldsymbol{p}^i_y \\ {}_a\boldsymbol{p}^i_z \end{pmatrix}. \tag{1.4}$$

Distance The Euclidean "squared distance" $d(\boldsymbol{p}^1, \boldsymbol{p}^2)$ between two points $\boldsymbol{p}^1$ and $\boldsymbol{p}^2$ is given by the well known formula

$$d(\boldsymbol{p}^1, \boldsymbol{p}^2) = (\boldsymbol{p}^1_x - \boldsymbol{p}^2_x)^2 + (\boldsymbol{p}^1_y - \boldsymbol{p}^2_y)^2 + (\boldsymbol{p}^1_z - \boldsymbol{p}^2_z)^2. \tag{1.5}$$

This notation is extended to the weighted distance $d_W(a, b)$ between two elements a and b in a given metric space (i.e., a space equipped with a distance function), with $\boldsymbol{W}$ the "weighting" matrix of the distance function. If the space is a linear vector space, a and b are vectors (so, they are denoted by the boldface symbols $\boldsymbol{a}$ and $\boldsymbol{b}$) and the weighted squared distance reduces to the matrix expression

$$d_W(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a}^T \boldsymbol{W} \boldsymbol{b}. \tag{1.6}$$

An example where this weighting is used could be the following: if $\boldsymbol{a}$ and $\boldsymbol{b}$ represent two sets of joint velocities of a serial robot, and $\boldsymbol{W}$ is the inertia matrix of the robot expressed in joint coordinates, then the $\boldsymbol{W}$-weighted distance between both sets of joint velocities is $d_W(\boldsymbol{a}, \boldsymbol{b}) = \frac{1}{2}\boldsymbol{a}^T \boldsymbol{W} \boldsymbol{b}$, which has the physical dimensions of kinetic energy.
$\boldsymbol{a}^T$ denotes the transpose of the column vector $\boldsymbol{a}$:

$$\boldsymbol{a}^T = \begin{pmatrix} \boldsymbol{a}_1 \\ \vdots \\ \boldsymbol{a}_n \end{pmatrix}^T = \begin{pmatrix} \boldsymbol{a}_1 & \ldots & \boldsymbol{a}_n \end{pmatrix}. \tag{1.7}$$

Dimensions Robots live in the *three-dimensional* Euclidean space, E^3. Note however, that this notion of three-dimensionality comes from looking at the *points* in this space. On the other hand, a *rigid body* in E^3 has *six* degrees of motion freedom, and mobile robots can most often be modelled as living on a plane, i.e., with only *three* degrees of motion freedom. Hence, saying that a particular model is "3D" or "2D" could lead to some confusion. This text avoids this confusion, by using these terms to refer to the *motion degrees of freedom* of an object, and not to the dimension of the space the object lives in. If the Euclidean space is equipped with an *orthogonal reference frame* with respect to which the coordinates of all entities living in the Euclidean space are expressed, we call this Euclidean space a *Cartesian space* (named after the French philosopher René Descartes, 1596–1650).

Kinematic joints The robotic devices discussed in this text are mechanical constructions whose motion degrees of freedom are always generated by a limited set of *joints*. Only four types of joints will be used: (i) *revolute* joints (denoted by the symbol "R," with one degree of rotational motion freedom; revolute joints are also called *rotational* joints), (ii) *prismatic* joints ("P," with one degree of translational motion freedom; prismatic joints are also called *sliding* joints), (iii) *spherical* joints ("S," with three degrees of rotational motion freedom), and (iv) *universal* joints (also called "Hooke" joints (Fig. 1.5), "U," with two degrees of rotational motion freedom).

Position and orientation of a rigid body A point has a unique *position* with respect to a chosen reference point (or frame). A *rigid body* is a collection of points, that always keep the same relative position with respect to each other. A *frame*, rigidly fixed to a rigid body, can serve as a mathematical abstraction of that rigid body; the position of the frame's origin, and the orientation of the frame's axes, uniquely determine the position and orientation of the rigid body to which the frame is attached. This text often uses the words "pose" and "frame" interchangeably as shorthands for "position and orientation."

Base and end effector frames Serial and parallel manipulators have a *base* and an *end effector*: the base is the immobile part of the robot, rigidly fixed to the world; the end effector is the extremity of the last link, to which tools can be attached (hence, it is often called *mounting plate* too). In the geometric model of the manipulator, base and end effector are represented by the reference frames $\{bs\}$ and $\{ee\}$, respectively.

References for this Chapter

[1] J. Agulló, S. Cardona, and J. Vivancos. Kinematics of vehicles with directional sliding wheels. *Mechanism and Machine Theory*, 22(4):295–301, 1987.

[2] K. Čapek. *R. U. R.* Les Cahiers Dramatiques. Théâtre et Comœdia Illustré, 1924.

[3] O. Fisher. *Theoretische Grundlagen für eine Mechanik der lebender Körper*. B. G. Teubner, Leipzig, Germany, 1906.

[4] V. E. Gough. Contribution to discussion of papers on research in automobile stability, control and tyre performance. In *Proc. Auto Div. Inst. Mech. Eng.*, pages 392–394, 1956-1957.

[5] R. Holmberg and O. Khatib. Development of a holonomic mobile robot for mobile manipulation tasks. In *International Conference on Field and Service Robotics*, 1999.

[6] F. G. Pin and S. M. Killough. A new family of omnidirectional and holonomic wheeled platforms for mobile robots. *IEEE Trans. Rob. Automation*, 10(4):480–489, 1994.

[7] D. Stewart. A platform with six degrees of freedom. *Proc. Instn Mech. Engrs*, 180-1(15):371–386, 1965.

[8] M. Wada and S. Mori. Holonomic and omnidirectional vehicle with conventional tires. In *IEEE Int. Conf. Robotics and Automation*, pages 3671–3676, Minneapolis, MN, 1996.

[9] H. West and H. Asada. Design of ball wheel mechanisms for omnidirectional vehicles with full mobility and invariant kinematics. *Trans. ASME J. Mech. Design*, 119:153–161, 1997.

Chapter 2

Model-based uncertainty

2.1 Introduction

This course focuses on *model-based* robot control, i.e., the sensing, planning and control modules all rely on explicit mathematical models of the robot system and its interactions with the environment. Models are used to *predict* the behaviour of the robot; the difference between these predictions on the one hand, and the sensor measurements on the other hand, provide information about how to *adapt* the models. Adaptation is necessary because, obviously, these models are always incomplete or inaccurate. In other words, they suffer from *uncertainty*, or *incomplete information.* This Chapter gives the fundamentals of how the *Bayesian probability* approach copes with this uncertainty.

Roughly speaking, there exist three levels of uncertainty in the system models, that give rise to three corresponding levels of increasingly complex *estimation* or *identification* algorithms:

1. *Parameter estimation.* In this sort of problems, the robot has a parameterized model of the system, and its goal is to estimate these parameters. Examples are: estimation of the geometric and/or dynamic parameters of a robot arm; estimation of the motion of an object that the robot perceives through its sensors; estimation of the geometry of the environment, using lines as primitive model building blocks. Typical algorithms in this category are the *Kalman Filter* and the *least-squares estimators.*
2. *Pattern recognition.* The robot has a model of a particular object it has to recognize and localize in its environment. That is, it has to compare the model to the continuous stream of measurements, and find the best match. Examples are: finding faces in camera images; finding the position and orientation of workpieces in an assembly cell; finding a particular room in a building. The *Hidden Markov Model* algorithm is one of the best-known representatives in this category (and is also the basis of speech recognition).
3. *Model building.* Instead of requiring the human programmer to come up with the models that the robot needs, one could use the robot and its sensors to produce the model. For example, a mobile robot driving around in a building (under human supervision, or autonomously using simple obstacle avoidance skills) should be able to produce a (at least topologically correct) map of the building. The *EM algorithm* is the algorithm of choice for this job.

This text also tries to make clear the distinction between, on the one hand, *updating of information* based on new measurements, and, on the other hand, the *process of decision making* that is based on this information.

2.2 Knowledge representation

The human programmers must store their knowledge about a task in a computer-processable form. This *knowledge representation* can be done by means of a *mathematical model*, or by giving a set of *statements* (also called *propositions*) about the world. (These propositions themselves, of course, could be derived from a mathematical model.) Knowledge representation is a multi-stage process: (i) identify the knowledge needed to solve the problem (this is the most difficult step, invariably performed by the human), (ii) select a *language* in which that knowledge can be represented, and (iii) write down the knowledge in that language, [11, p. 107]. The use of propositions is commonplace in classical logic: each proposition says something about the world, and is either true ("1") or false ("0"). Probability theory, and many of its "competitors," use the whole interval $[0, 1]$ instead of the binary set $\{0, 1\}$. Two fundamental *structures* exist in knowledge representation:

1. *State.* The state is the set of parameters that contain all information needed to predict the future evolution of the controlled system. This means that the past can be forgotten: the exact way how the system arrived in its current state does not influence its future evolution. A system with this property is also often called a *Markov process.* The Markov property is very interesting in real-time control, since it allows to reduce the dimension of the model that represents the system and which increases the efficiency of inference (i.e., the model update algorithms). On the other hand, a Markov process cannot "undo" any estimates. For example, a mobile robot that builds a map of an indoor environment must be able not only to adapt the estimate of its current position with respect to the currently observed "landmarks" in its environment, but also to adapt drastically its map itself (e.g., change the topological connection of corridors or rooms), whenever it has unexpectedly seen some landmarks more than once, [43].

   The state of a system is not always directly measurable: the *hidden states* must be "observed" ("estimated") from (i) the measurements, and (ii) a model that links the measurable quantities to the hidden and directly observable states. This estimation is a prime example of the use of *inference* in robotics.

2. *Networks.* If the state of the world is too big, one often divides it into smaller parts whose information content is only *weakly coupled.* These sub-parts make up a network. The A.I. literature gives many names to some similar *network* (or *graph*) structures of propositions: belief network [29], Bayes net, causal graph [21], causal network [44], influence diagram, relevance diagram, recursive model [45], semantic net, [20]. A network consists of *edges* and *nodes.* Each node is a proposition, each edge denotes dependence of the "truth values" (binary logic, probability, or other) of the propositions at both ends of the edge. A *directed* edge denotes *causal* influence: the node from where the directed edge starts is the physical cause of the node where the edge ends. In general, an edge (directed or undirected), or a set of edges, represents "probabilistic dependence" of connected nodes, i.e., knowing something about the proposition in one node gives information about the proposition in the other node. For example, the fact that one "eye" in a robotic stereo camera has detected a bright source of light, increases the probability for the other eye to detect the bright spot too, but it does not *cause* the detection.

A network allows two different kinds of reasoning:

1. *Causal reasoning.* This is reasoning from causes to consequences, i.e., following the arrows in the network. It *predicts* the outcome of the system when the inputs to it are known.

2. *Evidential reasoning* (or, *diagnostic reasoning*). This is reasoning from consequences to causes, i.e., the basis of "plausible inference," or "reasoning under uncertainty": observing a certain "symptom" makes a (number of) "hypotheses" more or less probable. "Plausible reasoning" means several things: it should correspond to what a human would call common sense; it should use all information it has, and not invent any data; it should be consistent, in the sense that the same conclusion should be drawn from the same data, irrespective of the reasoning steps in between and independent of which person or computer agent does the reasoning

(provided that different agent and persons do not have different background information that is relevant to the problem at hand!).

Figure 2.1 presents a simple example. The robot brings its probe in contact with the environment (only the probe is shown in the figure!). *If* the robot knows with which geometric "feature" (i.e., "I," "II" or "III") the probe is in contact, it can deduce (i.e., reason causally) what next contact transition it can expect: falling off an edge to the left or to the right, or bouncing against a "wall" in either direction. On the other hand, if the robot has only information about which contact transitions it has already detected, it must make plausible assumptions (reason evidentially) about the actual geometric feature.

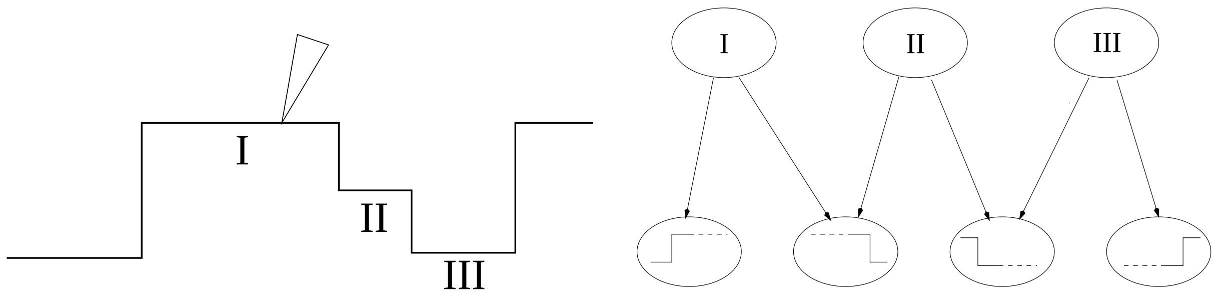


Figure 2.1: Left: a probe contacts the environment on one of its horizontal features ("I," "II" or "III"). Right: the features "causally" determine what contact transitions are possible.

More important than the fact that connected edges in a network correspond to correlated propositions, is the fact that the network models *conditional independence*, i.e., *given* the value of a node in the network, the values of all its "pre-decessor" nodes that are connected to each other only through that node, are determined by the value of that node, and *not* by the value of the other pre-decessor nodes.

A network puts *structure* on the knowledge about the system; it factors a problem in sub-problems, or, in other words, it models *conditional independence* of parameters in a system, e.g., [1, 30, 32]. In the words of Judea Pearl, [30, p. 350]: "That structure is more important than numbers in probability theory needs to be taught to all users of probability." The network models the structure in the knowledge the robot has about the task; but it is still most often the human user who designs the network.

During a sensor-based robot task, the values of the different parameters in the network change, due to new information from the sensors. In general, nodes and/or edges must also be added or deleted. However, this text does not discuss this problem, nor does it discuss the techniques to limit the computational burden of updating large networks; see e.g., [4, 22, 38] for some pointers to the literature.

2.3 Sources of uncertainty

Uncertainty corresponds to *incomplete information*, i.e., the truth values of the propositions that model the world cannot be determined unambiguously. Uncertainty has many sources:

1. *Poor sensors.* One should try for oneself to imagine what the sensing capabilities of robots are: a computer program investigating the pixels in a camera image has an efficiency and a typical "field of view" comparable to a person who looks at a scene in the mist and through a small peephole; a robot arm equipped with a force sensor gets information similar to a blind and deaf person carrying a white stick; a mobile robot equipped with a ring of ultrasonic sensors receives nothing more but a stream of echos similar to those heard by a

blind person who throws pebble stones in his environment at regular intervals in time and space. Not very rich information at all...

2. *Poor models.* Every model is only an *approximation* of the real world, a compromise between accuracy on the one hand, and efficiency of modelling and sensor processing on the other hand. For example, make a mental model of your bicycle, and then compare that mental model to the real one, just to find out that you undoubtedly forgot many details. But most probably, all relevant *functional* details will be there.
3. *Poor frame.* The word "frame" (or, "context") is used here in its A.I. sense: it represents all a priori, background or default knowledge that the robot possesses when executing a task. Frames are very difficult to keep complete, updated, and consistent, e.g., [39]. One of the largest problems in man–machine interaction is to have the machine recognize the context in which the human users gives their commands to the machine, or in which they expect information feedback from the machine.
4. *Poor experience.* Even if the model is sufficiently rich for the current task, the robot has not always had the chance to gather information about all relevant parameters in the model.

There exist basically three (complementary!) ways to deal with uncertainty, [37]:

1. *Reduce it physically.* This is what is done in industrial "hard" automation: all parts are fed to the robot at exactly known locations. This requires a very expensive set of peripheral tools and long set-up times, while drastically reducing the flexibility.
2. *Tolerate it.* This means that the robot controller must be *robust* against deviations between the real and the expected evolution of the system. This is the usual approach at the lower control levels, because uncertainties are "small" and on a continuous scale. Typical examples are: inaccurate knowledge about the friction in a motor, or about the exact direction of the contact normal.
3. *Represent it and reason about its consequences.* This is the approach to be taken by high-level "intelligent" robots, in order to cope with "large" and discrete uncertainties. Typical examples are: there is a door on the left or there is a staircase; the "landmark" that has just been observed corresponds to the exit towards the secretary or rather to the exit towards the wardrobe.

2.4 Information

Bayesian probability theory represents information about a given system by means of a *probability density function* ("pdf") over that system's state space. Intuitively speaking, a probability distribution with one or more sharp peaks contains more information on the state than one in which the probability mass is more evenly spread out. Good [13] used a coordinate-free way to capture this qualitative intuition about information in a quantitative measure, i.e., a set of numbers and/or real functions (Subsection 2.4.1). These are probability distributions (Section 2.4.2), of which the *Gaussian distributions* (Section 2.4.3) are the best-known example. Good's approach leads to the *global*, *scalar* measures of Shannon entropy (Section 2.4.4) and relative entropy (Section 2.4.5), and the *local*, *matrix* measure of the Fisher information (Section 2.4.6).

2.4.1 Foundations of information

Let's formally denote the information about model M that can be derived from the information E, and given the "background" ("context") information C by the notation $I(M{:}E|C)$. Good [13] proposed the following structural properties for $I(M{:}E|C)$:

1. $I(M{:}E \text{ AND } F|C) = f\{I(M{:}E|C), I(M{:}F|E \text{ AND } C)\}$.

2. $I(M{:}E \text{ AND } M|C) = I(M|C)$. If one knows already everything about M, other information cannot add anything anymore.
3. $I(M{:}E|C)$ is a strictly increasing function of its arguments: if the information content of one of the parameters of I increases, the information I increases too.
4. $I(M_1 \text{ AND } M_2{:}M_1|C) = I(M_1{:}M_1|C)$ if M_1 and M_2 are mutually irrelevant pieces of information.
5. $I(M_1 \text{ AND } M_2|M_1 \text{ AND } C) = I(M_2|C)$. Considering the information contained in M_1 doesn't increase the total information if this information was already incorporated.

Good found that these specifications lead to *many alternatives* for the representation of information. However, he also proved that, *if* information $I(M|C)$ is represented by a *measurable* function[1] $p(M|C)$, *then* composition of information becomes *additive* if and only if I is any function of $\log(p)$. The simplest choice being, of course, $I = \log(p)$. Hence:

- *Probability distributions* are those measurable functions that are *arbitrarily* normalized to have a *unit* area under their curve. So probability distributions are a natural choice to represent information (or, at the same time, uncertainty, which is the (partial) lack of information). And Gaussian distributions have become very popular, because of their attractive computational properties (Sect. 2.4.3).
- The choice $I = \log(p)$ is the rationale behind the abundance of logarithms in statistics (for example in the information measures discussed in the following Subsections), because addition is the *natural* operator on the space of these logarithms.

The logarithm of a probability distribution is in general not representable by a small set of real numbers. Hence, several derived scalar or matrix measures have been developed over the years. The following Subsections discuss three of them: Shannon entropy, Kullback and Leibler's relative entropy, and the Fisher Information matrix. But first, the next Subsections summarize the properties of (Gaussian) probability distributions.

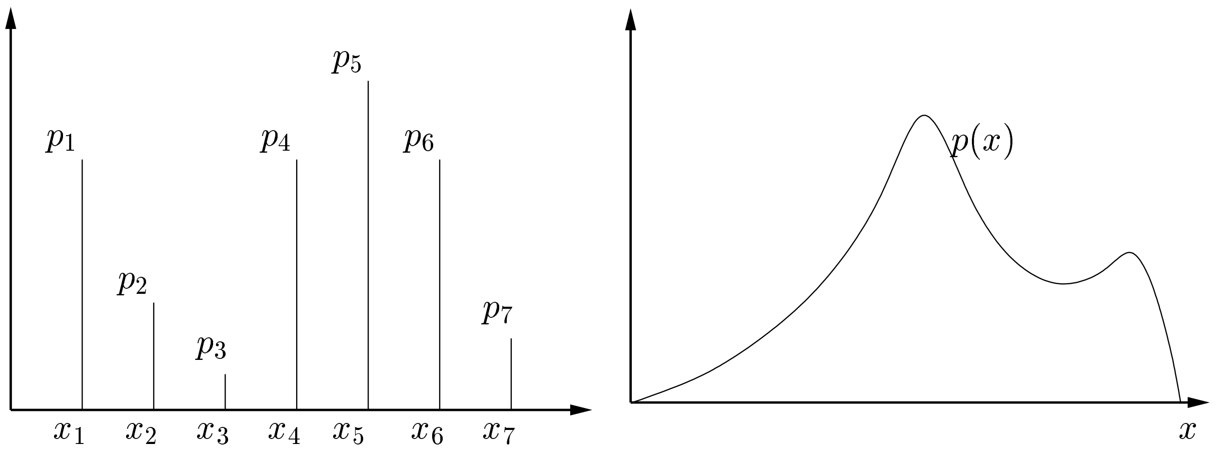


Figure 2.2: Examples of discrete and continuous probability distributions.

2.4.2 Probability distributions

This text follows a model-based approach to "spatial" robotics. Hence, the *geometric and/or physical parameters* of the robot, its sensors, its tools, the objects in the environment, the relative positions and orientations of objects,

[1]Loosely speaking, this means that calculation of information requires no involved mathematical technicalities.

their relative velocities, the physical forces acting on some of them, etc., are the basic building blocks to represent knowledge. Representing uncertainty in this parameter-based approach is quite straightforward: the value of a parameter is not exactly known but given by a *probability distribution*. Figure 2.2 gives examples of discrete probabilities, and of a continuous probability distribution (also called a *probability density function* ("pdf"), or *probability densities* for short). The interpretation of these probability distributions is exactly as in classical statistics:

1. *Discrete case*: the *sum* of the probabilities assigned to different values is the probability that the exact value of the parameter is one of these different values.
2. *Continuous case*: the *integral* of the probabilities lying in a given range is the probability that the exact value of the parameter lies in this range.

The paragraphs above discussed the case of a *parameter* in a geometric or physical model. In that case it is straightforward to assume that the parameter could take a value in a (continuous or discrete) range of possible alternatives. The case of *propositions* is a bit less obvious: what would be the interpretation of a probability distribution on the statement that "*I'm 80% sure that this door is the exit to the secretary*"? The (still controversial!) answer is given by the concept of a *second-order probability distribution*, [5, 31]: the *mean value* of the distribution lies at 0.8, and the spreading of the distribution is a measure for how uncertain one is about that probabilty value "0.8." For example, one could have come up with the probability just as the result of a guess (large spread in the distribution), or, on the other hand, as the result of consulting a map and having found a large number of corresponding "landmarks" (low spread).

The discrete and continuous cases do not just differ in the words "sum" and "integral" used in their interpretation given in the previous Section. Taking the integral of a fuction $f(x)$ over a subset A of a parameter space involves the product of *two* things, (i) the parameter values $f(x)$ and (ii) the *measure* dx of the parameter space:

$$\int_A f(x)\, dx. \tag{2.1}$$

That measure dx represents the "parameter density" at the given subset, i.e., the amount ("mass") of parameters contained around x. Measures are not just used for notational purposes (i.e., to denote the variables over which to integrate), but they are examples of so-called *differential forms*. An n-dimensional differential form maps n tangent vectors to a real number. The tangent vectors form the edges of a "parallellepipedum" of volume in the parameter space; the result of applying the differential form on this parallellepipedum is the amount of volume enclosed in it. Measures have their own transformation rules when changing the representation (change of coordinates, change of physical units, etc.): changing the representation changes the coordinates of the tangent vectors, and hence also the mathematical representation of the differential forms should change, in order to keep the enclosed volume *invariant*. An invariant measure puts *structure* on the parameter space. It can also be interpreted as the generalized form of a *uniform* probability distribution: with this distribution, each unit of volume in parameter space is equally probable.

The concept of an invariant measure is important for parameter spaces that are fundamentally different from $\mathbb{R}^n$, i.e., that have a different *structure* than $\mathbb{R}^n$, even though they may have n real numbers as coordinates. The best-known example is probably the measure used to calculate the surface integral over a sphere: depending on whether one uses Cartesian x, y, and z coordinates, or spherical r and θ coordinates, the measure changes, [42]:

$$\int_A \{f(x, y, z)\}\ \{dx\ dy\ dz\} = \int_A \{f(r, \theta, \phi)\}\ \{r\ dr\ \sin(\theta) d\theta\ d\phi\}\,. \tag{2.2}$$

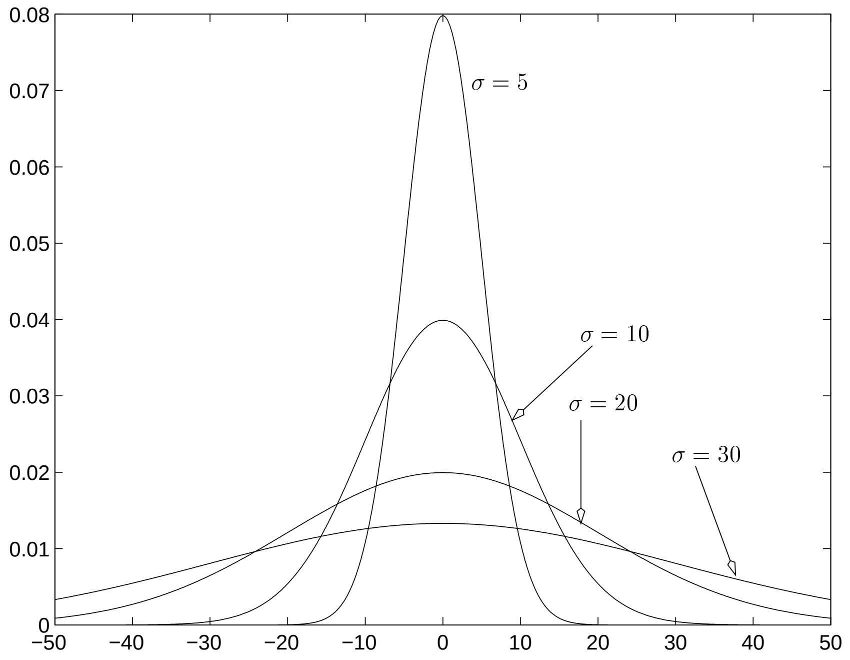


Figure 2.3: Gaussian probability distributions, with mean 0 and different standard deviations σ.

2.4.3 Gaussian probability distribution

A general probability distribution as in Fig. 2.2 has a quite arbitrary shape, and hence one needs a lot of parameters to represent it. The special class of *Gaussian probability distributions* needs only *two* parameters: the *mean* μ and the *standard deviation* σ, Fig. 2.3. Gaussian distributions have a lot of interesting properties from a *computational* point of view, which will become clear in the following Sections. A Gaussian distribution is symmetric around its mean; a larger standard deviation implies more uncertainty about the mean. Gaussians are often also called *normal* distributions, and are denoted by $\mathcal{N}(\mu, \sigma)$. The mathematical representation of a Gaussian with mean μ and standard deviation σ is:

$$\mathcal{N}(\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}. \tag{2.3}$$

σ^2 is called the *covariance* of the distribution. The Gaussian distributions above are *univariate distributions*, i.e., they are function of one single parameter x. Their *multivariate* generalizations have the form

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{P}) = \frac{1}{\sqrt{(2\pi)^n}\ ||\boldsymbol{P}||^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{P}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\}. \tag{2.4}$$

$\boldsymbol{x}$ is an n-dimensional vector, $\boldsymbol{\mu}$ is the vector of the mean (*first moment*, or "*expected value*") of $\boldsymbol{x}$:

$$\boldsymbol{\mu} = \int_{-\infty}^{\infty} dx_1 \ldots \int_{-\infty}^{\infty} dx_n \left\{\boldsymbol{x}\, p(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{P})\right\}. \tag{2.5}$$

$||\boldsymbol{P}||$ is the *two-norm* (i.e., the square root of the largest eigenvalue of $\boldsymbol{P}^T\boldsymbol{P}$, [12]) of the *second moment*, or *covariance matrix* $\boldsymbol{P}$:

$$\boldsymbol{P} = \int_{-\infty}^{\infty} dx_1 \dots \int_{-\infty}^{\infty} dx_n \, \left\{ (\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^T \, p(\boldsymbol{a}|\boldsymbol{\mu}, \boldsymbol{P}) \right\}. \tag{2.6}$$

Note that the term $(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^T$ in the above equation is a *matrix*, while the term $(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{P}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$ in Eq. (2.4) is a *number*. It can be shown that this number has all properties of a *distance* on the parameter space of $\boldsymbol{x}$; in other words, the inverse of the covariance matrix $\boldsymbol{P}$ is a *metric* on that space, [34], and hence determines *structure* on the space.

2.4.4 Shannon entropy

Claude Elwood Shannon (1916–), [6, 40, 41], presented a *scalar* "average" (or "expected") measure to quantify the quality of communication channels, i.e., their capacity to transmit information. However, his measure also qualifies as a fully general information measure, as was first explained by Jaynes [14]. Shannon's measure is known under the name of *entropy*, because it models the similar concept with the same name in thermodynamics: the higher the entropy of a thermodynamic system, the higher our uncertainty about the state of the system, or, in other words, the higher its "disorder." Note that entropy is a "subjective" feature of the system: it represents the knowledge (or uncertainty) that the *observer* has of the system, but it is not a physical property of the system itself. However, it is "objective" in the sense that each observer comes to the same conclusion when given the same information.

The following paragraphs discuss only discrete probability distributions, since they are more intuitive than continuous distributions. Assume the parameter x takes one of the values from a set $\{x_1, \dots, x_n\}$, with $p(x) = \{p_1, \dots, p_n\}$ the corresponding probability distribution. The entropy $H(p)$ of the probability distribution $p(x)$ is derived from the following three desiderata:

Axioms for entropy

I	H is a *continuous* function of p.
II	If all n probabilities p_i are equal (and hence equal to $1/n$, because they have to sum to 1), the entropy $H(1/n, \dots, 1/n)$ is a *monotonically increasing* function of n.
III	H is an *invariant*, i.e., the uncertainty should not depend on how one orders or groups the elements x_i.

The first and second specifications model our intuition that (i) small changes in probability imply only small changes in entropy, and (ii) our uncertainty about the exact value of a parameter increases when it is a member of a larger group. The third desideratum is represented mathematiclly as follows: the entropy H obeys the following *additive composition law*:

$$\begin{aligned} H(p_1, \dots, p_n) = & H(w_1, w_2, \dots) \\ & + w_1 H(p_1|w_1, \dots, p_k|w_1) \\ & + w_2 H(p_{k+1}|w_2, \dots, p_{k+m}|w_2) + \dots, \end{aligned} \tag{2.7}$$

here w_1 is the probability of the set $\{x_1, \dots, x_k\}$, w_2 is the probability of the set $\{x_{k+1}, \dots, x_{k+m}\}$, and so on; $p_i|w_j$ is the probability of the alternative x_i *if* one knows that the parameter x comes from the set that has probability w_j. For example, assume that x comes from a set of three members, with the alternatives occurring with probabilities $1/2, 1/3$ and $1/6$, respectively. If one then groups the second and third alternatives together (i.e., $w_1 = p_1 = 1/2$, the probability of the set $\{x_1\}$, and $w_2 = p_2 + p_3 = 1/3 + 1/6 = 1/2$, the probability of the set $\{x_2, x_3\}$), the composition law gives $H(1/2, 1/3, 1/6) = H(1/2, 1/2) + \frac{1}{2}H(2/3, 1/3)$ since $2/3$ and $1/3$ are the probabilities of x_2 and x_3 within the set $\{x_2, x_3\}$.

The three above-mentioned *axioms* suffice to derive an analytical expression for the entropy function $H(p)$. The first axiom implies that it is sufficient to determine $H(p)$ for *rational* values $p_i = n_i / \sum_{j=1}^{n} n_j$ (with n_j integer numbers) only; the reason is that the rational numbers are a *dense* subset of the real numbers. One then uses the composition law to find that $H(p)$ can be found from the uniform probability distribution $P = (1/N, \ldots, 1/N)$ over $N = \sum_{i=1}^{n} n_i$ alternatives. Indeed, the composition law says that the entropy $H(p)$ is equal to the entropy $H(P)$, because in P one can group the first n_1 alternatives, the following n_2 alternatives, and so on, which reduces to the original distribution. For example, let $n = 3$ and $(n_1, n_2, n_3) = (3, 4, 2)$ such that $N = 3 + 4 + 2 = 9$; denoting $H(1/N, \ldots, 1/N)$ by $H(N)$ yields

$$H\left(\frac{3}{9}, \frac{4}{9}, \frac{2}{9}\right) + \frac{3}{9}H(3) + \frac{4}{9}H(4) + \frac{2}{9}H(2) = H(9).$$

In general, this could be written as

$$H(p_1, \ldots, p_n) + \sum p_i H(n_i) = H\left(\sum n_i\right). \tag{2.8}$$

The special case of all n_i equal to the same integer m gives

$$H(n) + H(m) = H(mn).$$

A solution to this equation is given by $H(n) = K \ln(n)$, with $K > 0$ because of the monotonicity rule. All this yields the following expression for the entropy:

$$H(p_1, \ldots, p_n) = K \ln\left(\sum n_i\right) - K \sum p_i \ln(n_i),$$

or:

Fact-to-Remember 1 (Entropy of a probability distribution)

$$H(p_1, \ldots, p_n) = -K \sum p_i \ln(p_i). \tag{2.9}$$

The minus sign makes that entropy increases when uncertainty increases. The constant K has no influence: it is nothing but a factor that sets the scale of the entropy. The entropy need not be a monotonically decreasing function of the amount of information received: entropy can increase with new information ("evidence") coming in, if this new information contradicts the previous assumptions. Note also that the uncertainty in many *dynamic* systems increases naturally over the time period that no new information is received from the system: the probability distributions "flatten out" and hence the entropy increases.

2.4.5 Kullback-Leibler divergence

Equation (2.9) suggests that the entropy for a continuous probability distribution is $H(p(x)) = -K \int p(x) \ln(p(x))\, dx$. However, this extrapolation neglects the important difference between discrete and continuous distributions: the probability mass enclosed in an interval does not only depend on the magnitude of the probability distribution in that interval, but also on the amount of parameter "volume" in that interval. The only way to get an entropy that *is* invariant is to consider the *relative entropy* of *two* probability distributions, here represented by the densities $p(x)$ and $m(x)$ over the same measure dx:

$$H\big(p(x){:}m(x)\big) = -\int \log\left(\frac{p(x)}{m(x)}\right) p(x)dx. \tag{2.10}$$

This scalar was called the *Kullback-Leibler divergence*, (after the duo that first presented it, [18], [19]), or also *mutual entropy*, or *cross entropy*, of both probability measures $p(x)$ and $m(x)$, [18, 35]. It is a (coordinate-independent) measure for how much information one needs to go from the probability distribution $m(x)$ to the probability distribution $p(x)$. As Shannon's entropy, $H\big(p(x){:}m(x)\big)$ is a *global* measure, since all information contributions $\log(p(x)/m(x))$ are weighted by $p(x)dx$, and then added together.

2.4.6 Fisher Information

The relative entropy $H\big(p(x){:}m(x)\big)$ of Eq. (2.10) is *not* a metric, since it is not symmetric in its arguments: $H\big(p(x){:}m(x)\big) \neq H\big(m(x){:}p(x)\big)$. Rao [34] was the first to come up with a real metric on a manifold $\mathcal{M}_\Sigma$ of probability distributions $p(x,\Sigma)$ over the state space x and described by a parameter vector $\Sigma = \{\sigma_1,\dots,\sigma_n\}$. Define tangent vectors $v = (v^1,\dots,v^n)$ to the manifold $\mathcal{M}_\Sigma$ as follows:

$$v(x,\Sigma) = \sum_{i=1}^{n} v^i \frac{\partial l(x,\Sigma)}{\partial \sigma^i}, \qquad \text{with} \quad l(x,\Sigma) = \log\big(p(x,\Sigma)\big). \tag{2.11}$$

The v^i are the coordinates of the tangent vector in the basis formed by the tangent vectors of the *logarithms* of the σ-coordinates. (Section 2.4.1 explains why these logarithms show up all the time in statistics.) A metric $\mathcal{M}$ at the point p (which is a probability distribution) is a bilinear mapping that gives a real number when applied to two (logarithmic) tangent vectors v and w attached to p. Rao showed that the *covariance* of both vectors satisfies all properties of a metric. Hence, the elements g_{ij} of the matrix representing the metric are found from the covariance of the coordinate tangent vectors:

$$g_{ij}(\Sigma) = \int \big(\partial_i l(x,\Sigma)\big)\big(\partial_j l(x,\Sigma)\big) p(x,\Sigma)\, dx. \tag{2.12}$$

The matrix g_{ij} got the name *Fisher information matrix.* The covariance "integrates out" the dependency on the state space coordinates x, hence the metric is only a function of the statistical coordinates Σ. This metric is defined on the tangent space to the manifold $\mathcal{M}_\Sigma$ of the σ-parameterized family of probability distributions over the x-parameterized state space $\mathcal{X}$.

Kullback and Leibler already proved the following relationship between the relative entropy of two "infinitely separated" probability distributions Σ and $\Sigma + \epsilon v$ on the one hand, and the Fisher Information matrix $g_{ij}(\Sigma)$ on the other hand:

$$H(\Sigma{:}\Sigma + \epsilon v) = \frac{1}{2}\sum_{i,j} g_{ij}(\Sigma) v^i v^j + \mathcal{O}(\epsilon^2). \tag{2.13}$$

Hence, Fisher Information represents the *local* behaviour of the relative entropy: it indicates the rate of change in information in a given direction of the probability manifold (*not* in a given direction of the state space!).

2.5 Bayesian probability theory

The previous Section discussed properties of information, and indicated that probability density functions are "natural" representations for information. Good's approach does not say how to "calculate" with information, i.e., how to combine the information from different sources, or from the same source but collected at different time instants. This is the area of *probability theory.* This theory has many historical roots, and many approaches exist to explain its fundamental mathematical properties. The approach developed by Cox [7], and later refined by Jaynes [14], [15], [36] fits nicely into the information-based approach to inference presented in previous Sections. Cox and Jaynes looked at probability theory as the theory of how to deal with uncertain statements and hypotheses, and *not* as the more classical theory describing frequencies of random events. The former approach is currently known as *Bayesian* probability theory (e.g., [17]), in contrast to the latter "orthodox statistics" à la Fisher, [10]. Cox [7] starts from only three functional relationships to describe the *structure* that operations in uncertain reasoning should obey:

1. What we know about two statements A and B should be a smooth function of (i) what we know about A, and (ii) what we know about B *given* that A is taken for granted. In functional form this becomes:

$$p(A, B) = f\{p(A), p(B|A)\}, \tag{2.14}$$

   with $p(A)$ the probability of proposition A, $p(B|A)$ the conditional proposition of B given A, and f an as yet unspecified function. (This is the same starting point as Good's reasoning that led to the definition of information.) $p(A)$ can be a continuous probability distribution *function* (instead of a single number), for example if it represents the value of a parameter in the system under study. Cox then proves that relation (2.14) leads to the following form for f:

$$f\{p(A), p(B|A)\} = f\{p(A)\}^m f\{p(B|A)\}^m, \tag{2.15}$$

   for an arbitrary m and an arbitrary f. Hence, it turns out that the historical literature in statistics had already fixed the choices $m = 1$ and $f(u) = u$, not out of necessity but most probably just for computational convenience.

2. The negation of the negation of a proposition A is equal to the proposition A. This means that a function g must exist such that:

$$g\Big(g\big(p(A)\big)\Big) = p(A). \tag{2.16}$$

3. The same function g should also satisfy the following law of logic:

$$g\big(p(A \text{ OR } B)\big) = g\big(p(A)\big) \text{ AND } g\big(p(B)\big). \tag{2.17}$$

These structural prescriptions are sufficient to derive the product rule $p(A \text{ AND } B|M) = p(A|M)p(B|A \text{ AND } M)$, and the additivity to one, $p(A) + p(\text{NOT } A) = 1$. ($M$ represents all available knowledge ("models") used in the inference procedure.)

Jaynes builds further on the approach by Cox, and states some assumptions (e.g., invariance) a bit more explicitly. In his unfinished manuscript [15], the basic rules of plausible reasoning are formulated as follows:

Axioms for plausible Bayesian inference

I	Degrees of plausibility are represented by real numbers.
II	Qualitative correspondence with common sense.
III	If a conclusion can be reasoned out in more than one way, then every possible way must lead to the same result.
IV	Always take into account all of the evidence one has.
V	Always represent equivalent states of knowledge by equivalent plausibility assignments.

These "axioms" are not accepted without discussion, e.g., [11, p. 241]:

- Assigning equivalent probabilities for equivalent states seems to assume that the modeller has "absolute knowledge," since one often doesn't know that states are equivalent.
- Many people state that representing probability by one single real number is not always possible or desirable.

It can be proved, [7, 15], in a way again very similar to the proof given for the entropy desiderata, that the above-mentioned axioms lead to the following well-known probability rules:

Bayesian calculus

Sum rule

$$\boxed{p(x+y|H) = p(x|H) + p(y|H).} \tag{2.18}$$

Product rule

$$\boxed{p(xy|H) = p(x|yH)p(y|H).} \tag{2.19}$$

Bayes' rule

$$\boxed{p(\text{Model}|\text{Data}, H) = \frac{p(\text{Data}|\text{Model}, H)}{p(\text{Data}|H)} p(\text{Model}|H).} \tag{2.20}$$

Bayes' rule follows straightforwardly from the product rule, since that rule is symmetric in both its arguments. Equation (2.20) suggestively uses the names "Data" and "Model," since this is the contents of these variables in many robotics inference problems. The most important thing to notice is that

Fact-to-Remember 2 (Bayes' rule represents model-based learning)
It expresses the probability of the Model, given the Data (and the background information H), as a function of (i) the probability of the Data when the Model is assumed to be known, and (ii) the probability of that Model given the background information.

The denominator in Bayes' rule is usually regarded as just a normalization constant, [24, p. 105]; it is independent of the Model, and predicts the Data given only the prior information. The term $p(\text{Model}|\text{Data}, H)$ is called the *posterior probability*; $p(\text{Data}|\text{Model}, H)/p(\text{Data}|H)$ is the *likelihood*; and $p(\text{Model}|H)$ is the *prior probability* of the hypothesis. The likelihood is *not* a probability distribution in itself; as the ration of two probability distributions with values between 0 and 1 it can have any positive real value.

Bayesian probability theory gives an axiomatically founded treatment of *all* structures and concepts needed in reasoning with uncertainty; that's why it is presented in this text. Classical ("orthodox") statistics gives some shortcuts for particular problems; this is convenient for specific implementations (such as many parameter identification applications) but it is seldom easy and intuitive to know what shortcuts are used in a given robotics task.

2.5.1 Optimality of information processing

The reasoning in Section 2.4.1 also allows to interpret Bayes' rule as a procedure to combine information from two sources *without loss of information*: the first source is the prior information already contained in the current state, and the second source is the new information added by the current measurement. This relationship is straightforward: take Bayes' rule for two models M_1 and M_2 that receive the same new $Data$:

$$p(M_1|Data) = \frac{p(Data|M_1)}{p(Data)}\, p(M_1),$$
$$p(M_2|Data) = \frac{p(Data|M_2)}{p(Data)}\, p(M_2).$$

Taking the logarithms of the ratio of both relationships yields

$$\log \frac{p(M_1|Data)}{p(M_2|Data)} = \log \frac{p(Data|M_1)}{p(Data|M_2)} + \log \frac{p(M_1)}{p(M_2)}. \tag{2.21}$$

The left-hand side is the information *after* the measurement; the right-hand side represents the information contributions of both sources. Hence, information "before" is equal to information "after" and Bayes' rule is optimal in this respect, [46].

2.5.2 Estimation—Hypothesis testing

Bayes' rule is the *general* basis underlying, for example, parameter estimation and hypothesis testing. But the literature describes many *specific* estimators, as special cases of Bayes' rule:

Maximum Likelihood (ML) : the maximum ("*mode*") of the likelihood function.

Maximum A Posteriori (MAP) : the mode of the posterior probability.

Least Squares (LS) : the minimum of the "squared error function," i.e., the function $(\hat{x}-x)^T(\hat{x}-x)$. It seems at first sight that this estimator involves no probability distributions on the parameter(s) x; but the squared error is proportional to the exponent of a Gaussian distribution with equal covariance in all parameters, which are independent; hence their maxima coincide.

Some of these estimators become equivalent in special cases:

- ML = MAP if the prior is noninformative, i.e., "flat."
- LS = ML, for indentically distributed and independent ("*iid*"), symmetric and zero mean distributions.

Another interesting property is that Bayes' rule with Gaussian distributions as inputs gives a Gaussian distribution as posterior, [23, p. 7]. Hence, using Gaussians to represent uncertainty reduces the computational complexity of the software enormously. Some less interesting properties are:

- General probability distributions require a lot of computing power.
- Although Bayes' rule is a consistent and fundamental way to perform inference, its results are only as good as its input. For example, if the Data comes from a sensor that has a *systematic error*, there is no way Bayes' rule can compensate for this, without having a model of the error available.
- MAP estimates do not have any fundamental status: MAP finds the maximum in the probability distribution, but does not take into account the "volume" ("invariant measure") of the parameter space over which the maximum is found. What matters is the probability "mass," i.e., the product of "volume" and "probability measure," [25]. Hence, the MAP can depend on the chosen coordinate representation.

Hypothesis tests Hypothesis tests are, in fact, the same things as parameter estimators, [24, p. 104]: estimating a parameter to have a certain value is the same problem as testing the hypothesis that it has that value against the hypotheses that it has other values. The mathematical formulation follows from dividing two instantiations of Bayes' rule—Eq. (2.20)—with two models "Model1" and "Model2":

$$\frac{p(\text{Model1}|\text{Data}, H)}{p(\text{Model2}|\text{Data}, H)} = \frac{p(\text{Data}|\text{Model1}, H)}{p(\text{Data}|\text{Model2}, H)} \, \frac{p(\text{Model1}|H)}{p(\text{Model2}|H)}. \tag{2.22}$$

Hence, the relative probability of two hypotheses not only depends on how well they predict the Data (i.e., the first term on the right-hand side; more complex models *always* fit the data better), but also on how *complex* the models are: the second term on the right-hand side is larger than 1 if Model1 is simpler than Model2, because simpler models have larger entropy and hence more probability to occur. This principle is often called *Occam's razor*, after (the Latin name of) the Englishman William of Ockham (1288–1348), who got famous (long after his death!) for his quotations "*Frustra fit per plura, quod fieri potest per pauciora* (It is vain to do with more what can be done with less) and "*Essentia non sunt multiplicanda praeter necessitatem*" (Entities should not be multiplied unnecessarily).

State estimation

The concept of a state (with and without "hidden" paramaters) is very important in robotics, and hence *state estimation* algorithms have been and still are a major research area. The literature describes three main classes of algorithms:

1. The *Kalman Filter* for on-line parameter estimation, e.g. [2, 16, 26], and its off-line cousin *weighted least-squares*. [28] shows how the Kalman Filter, Fig. 2.4, follows from Bayes' rule, when the system under consideration is *linear*, and the probability distributions are *Gaussians*.
2. The *Hidden Markov Model* (HMM) for pattern recognition, e.g., [3, 33]. A HMM models the *discrete transitions* ("jumps") between states in a finite state machine, and assigns probabilities to the transitions.
3. The *EM algorithm* for model building, e.g. [8, 27]. It is similar to the Kalman Filter, but much more general: the system can have nonlinear state space, and also the probabilities can be more general, i.e., non-Gaussian.

These algorithms estimate the parameters of a model iteratively, starting from some initial guess; the Kalman Filter is designed to work in real-time, on time-varying Markov systems, while the EM algorithm can handle more complicated and non-Markov inference problems, but needs more time and hence is not very appropriate to track time-varying systems. The complexity and properties of the HMM are somewhere in between. Each iteration consists of:

1. A *Prediction/Expectation* step, that finds the distributions of the unobserved states, given the known values for the observed states and he current estimate of the parameters. For time-varying systems, the "unobserved" parameters include the (observable and unobservable) parameters that undergo a stochastic time-variance.
2. A *Correction/Maximization* step, that re-estimates the parameters to those with maximum likelihood, under the assumption that he distribution found in the P/E step is correct. This step can be done efficiently in the Kalman Filter because it assumes that all uncertainties are represented by Gaussian distributions.

References for this Chapter

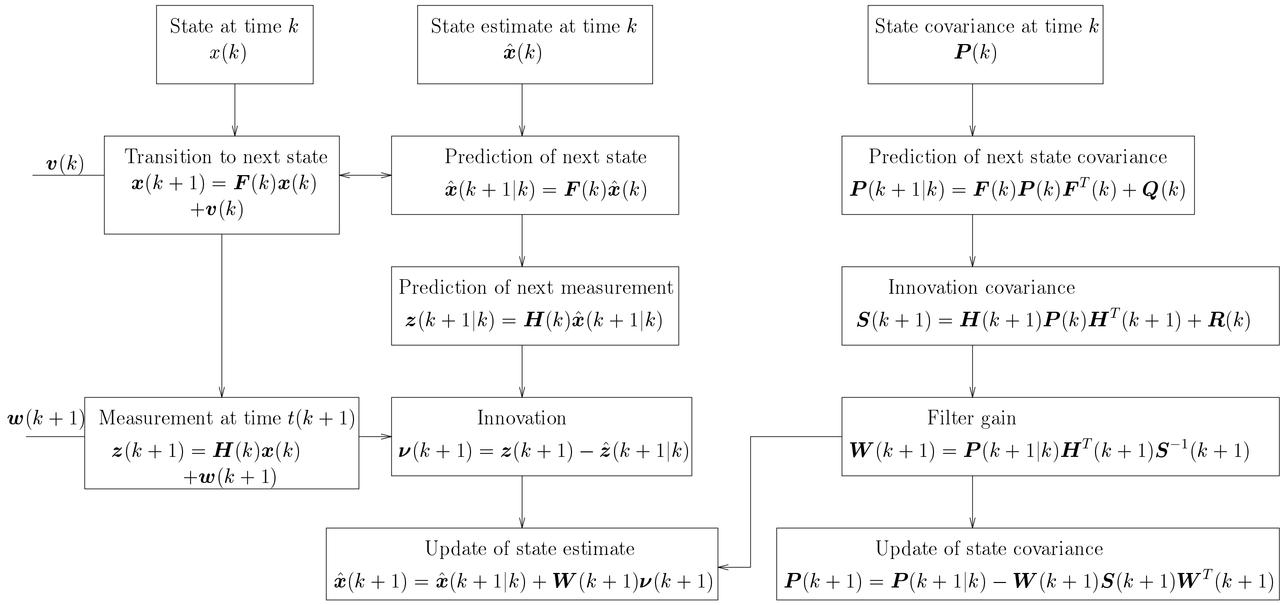

State at time k
$x(k)$
State estimate at time k
$\hat{\boldsymbol{x}}(k)$
State covariance at time k
$\boldsymbol{P}(k)$
$\boldsymbol{v}(k)$
Transition to next state
$\boldsymbol{x}(k+1) = \boldsymbol{F}(k)\boldsymbol{x}(k) + \boldsymbol{v}(k)$
Prediction of next state
$\hat{\boldsymbol{x}}(k+1|k) = \boldsymbol{F}(k)\hat{\boldsymbol{x}}(k)$
Prediction of next state covariance
$\boldsymbol{P}(k+1|k) = \boldsymbol{F}(k)\boldsymbol{P}(k)\boldsymbol{F}^T(k) + \boldsymbol{Q}(k)$
Prediction of next measurement
$\boldsymbol{z}(k+1|k) = \boldsymbol{H}(k)\hat{\boldsymbol{x}}(k+1|k)$
Innovation covariance
$\boldsymbol{S}(k+1) = \boldsymbol{H}(k+1)\boldsymbol{P}(k)\boldsymbol{H}^T(k+1) + \boldsymbol{R}(k)$
$\boldsymbol{w}(k+1)$
Measurement at time $t(k+1)$
$\boldsymbol{z}(k+1) = \boldsymbol{H}(k)\boldsymbol{x}(k) + \boldsymbol{w}(k+1)$
Innovation
$\boldsymbol{\nu}(k+1) = \boldsymbol{z}(k+1) - \hat{\boldsymbol{z}}(k+1|k)$
Filter gain
$\boldsymbol{W}(k+1) = \boldsymbol{P}(k+1|k)\boldsymbol{H}^T(k+1)\boldsymbol{S}^{-1}(k+1)$
Update of state estimate
$\hat{\boldsymbol{x}}(k+1) = \hat{\boldsymbol{x}}(k+1|k) + \boldsymbol{W}(k+1)\boldsymbol{\nu}(k+1)$
Update of state covariance
$\boldsymbol{P}(k+1) = \boldsymbol{P}(k+1|k) - \boldsymbol{W}(k+1)\boldsymbol{S}(k+1)\boldsymbol{W}^T(k+1)$

Figure 2.4: Computational scheme of the Kalman Filter, [2].

[1] R. G. Almond. *Graphical Belief Modeling.* Chapman & Hall, 1995.

[2] Y. Bar-Shalom and X.-R. Li. *Estimation and Tracking, Principles, Techniques, and Software.* Artech House, 1993.

[3] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite Markov chains. *Annals of Mathematical Statistics*, 37:1554–1563, 1966.

[4] B. P. Carlin and T. A. Louis. *Bayes and empirical Bayes methods for data analysis.* Chapman & Hall, London, England, 1996.

[5] P. Cheeseman. Probabilistic versus fuzzy reasoning. In L. N. Kanal and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 85–102, Amsterdam, The Netherlands, 1986. North-Holland.

[6] T. M. Cover and J. A. Thomas. *Elements of information theory.* Wiley, New York, NY, 1991.

[7] R. T. Cox. Probability, frequency, and reasonable expectation. *American Journal of Physics*, 14(1):1–13, 1946. Reprinted in [38, p. 353].

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[9] G. J. Erickson and C. R. Smith, editors. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1988.

[10] R. A. Fisher. *Statistical methods for research workers.* Oliver and Boyd, Edinburgh, Scotland, 13th edition, 1967.

[11] M. Ginsberg. *Essentials of Artificial Intelligence.* Morgan Kaufmann, San Mateo, CA, 1993.

[12] G. Golub and C. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, 1989.

[13] I. J. Good. A derivation of the probabilistic explanation of information. *J. Roy. Stat. Soc., Ser. B*, 28:578–581, 1966.

[14] E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106:620–630, 1957. Reprinted in [36, p. 6–16].

[15] E. T. Jaynes. Probability theory: The logic of science. Unfinished manuscript, `http://bayes.wustl.edu/etj.`, 1996.

[16] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME J. Basic Eng.*, 82:34–45, 1960.

[17] M. G. Kendall and A. O'Hagan. *Kendall's advanced theory of statistics. 2B: Bayesian inference.* Arnold, London, England, 1994.

[18] S. Kullback. *Information theory and statistics.* Wiley, New York, NY, 1959.

[19] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22:79–86, 1951.

[20] S. L. Lauritzen. *Graphical Models*, volume 17 of *Oxford Statistical Science Series.* Clarendon Press, 1996.

[21] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society, Series B*, 50(2):157–224, 1988. Reprinted in [38, p. 415].

[22] P. M. Lee. *Bayesian statistics: an introduction.* Oxford University Press, New York, NY, 1989.

[23] D. V. Lindley. *Introduction to probability and statistics from a Bayesian viewpoint. Vol. 1: Probability, Vol. 2: Inference.* Cambridge University Press, 1965.

[24] T. J. Loredo. From Laplace to supernova SN 1987a: Bayesian inference in astrophysics. In P. F. Fougère, editor, *Maximum Entropy and Bayesian Methods*, pages 81–142. Kluwer, Dordrecht, The Netherlands, 1990.

[25] D. J. C. MacKay. Information theory, inference and learning algorithms. Textbook in preparation. `http://wol.ra.phy.cam.ac.uk/mackay/itprnn/`, 1999.

[26] P. S. Maybeck. *Stochastic models, estimation, and control. Vol. 1.* Number 141 in Mathematics in science and engineering. Academic Press, New York, NY, 1979. Republished by Navtech Press, Arlington, 1994.

[27] G. J. McLachlan and T. Krishnan. *The EM algorithm and extensions.* Wiley, New York, NY, 1997.

[28] R. J. Meinhold and N. D. Singpurwalla. Understanding the Kalman-Filter. *The American Statistician*, 37:123–127, 1983.

[29] J. Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29:241–288, 1986.

[30] J. Pearl. Bayesian decision methods. In *Encyclopedia of artificial intelligence*, pages 49–56. Wiley Interscience, New York, NY, 1987. Reprinted in [38].

[31] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann, San Mateo, CA, 1988.

[32] D. Poole. Learning, bayesian probability, graphical models, and abduction. In P. Flach and A. Kakas, editors, *Abduction and Induction: essays on their relation and integration.* Kluwer, 1998.

[33] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[34] C. R. Rao. Information and the accuracy attainable in the estimation of statistical parameters. *Bulletin of the Calcutta Mathematics Society*, 37:81–91, 1945.

[35] C. C. Rodríguez. The metrics induced by the Kullback number. In J. Skilling, editor, *Maximum Entropy and Bayesian Methods*, pages 415–422. Kluwer, 1989.

[36] R. D. Rosenkrantz, editor. *E. T. Jaynes: Papers on Probability, Statistics and Statistical Physics.* D. Reidel, 1983. Second paperbound edition, Kluwer Academic Publishers, 1989.

[37] A. Saffiotti. An AI view of the treatment of uncertainty. *The Knowledge Engineering Review*, 2(2):75–97, 1987.

[38] G. Shafer and J. Pearl, editors. *Readings in Uncertain Reasoning.* Morgan Kaufmann, San Mateo, CA, 1990.

[39] M. Shanahan. *Solving the frame problem: a mathematical investigation of the common sense law of inertia.* MIT Press, Cambridge, MA, 1997.

[40] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[41] C. E. Shannon and W. Weaver. *The mathematical theory of communication.* University of Illinois Press, Urbana, IL, 1949.

[42] G. Strang. *Calculus.* Wellesley-Cambridge Press, Wellesley, MA, 1991.

[43] S. Thrun, D. Fox, and W. Burgard. Probabilistic mapping of an environment by a mobile robot. In *IEEE Int. Conf. Robotics and Automation*, pages 1546–1551, Leuven, Belgium, 1998.

[44] T. Verma and J. Pearl. Causal networks: Semantics and expressiveness. In T. S. Levitt, editor, *Uncertainty in Artificial Intelligence*, pages 352–359, Amsterdam, The Netherlands, 1988. North-Holland.

[45] N. Wermuth and S. L. Lauritzen. Graphical and recursive models for contingency tables. *Biometrika*, 72:537–552, 1983.

[46] A. Zellner. Optimal information processing and Bayes's theorem. *The American Statistician*, 42:278–284, 1988.

Chapter 3

Geometry of motion

3.1 Introduction

Robots are in the first place positioning devices that move rigid bodies around in all possible directions. Hence, good *knowledge* about the *structure* of rigid body positions and orientations is an important prerequisite for any *intelligent* robot controller. This Chapter introduces all relevant properties *without* using a *coordinate* representation. (Of course, some mathematical *notation* will be used to describe the physical model.) This discussion requires only a small amount of time and space, but it covers all basic properties that are needed in the rest of the book. A coordinate-free description, however, is not sufficient if one really wants to *work* with robots, instead of just *describing* their properties. In other words, coordinate representations are indispensable to *implement* the model in computer programs. Hence, the following Chapters describe the coordinate representations that are most commonly used in robotics, and compare their advantages and disadvantages.

Fact-to-Remember 3 (Basic ideas of this Chapter)
Extracting the properties of rigid body motion from everyday experience has taken mankind several hundreds of years. The major reason is that motions of rigid bodies obey fundamentally different properties than motions of points, i.e., the structure of rigid body motion is that of a curved space.

3.2 Rigid body motion

There is one very important complication about *rigid bodies* that makes them a bit more tedious than *points* in the Euclidean, three-dimensional space (denoted E^3): the geometry of rigid body motion is the geometry of *frames* in E^3, since the *pose* (i.e., position and orientation) of a rigid body is uniquely defined by the pose of a reference frame attached to the rigid body. This space of frames does not have the geometry of the familiar Euclidean space, mainly due to the fact that rotations and translations do not commute, i.e., executing a translation first and then a rotation yields a different motion than executing the rotation first. This Chapter tries to capture these essential properties of rigid body motion, without using coordinate representations. This is important for two reasons: (i) a given coordinate representation is only acceptable *if* it has these properties; and (ii) every extra mathematical property that the coordinate representation might suggest has no physical meaning. At this point it might be difficult to grasp the full meaning of these statements. But the rest of the book will be much easier to digest if you look back at these paragraphs from time to time. So, the message of this Chapter is important enough to be repeated in a

Fact-to-Remember 4 (Rigid body motion and coordinates)
Every possible coordinate representation of rigid body motion has to have the properties presented in this Chapter.
Every extra mathematical property that a possible coordinate representation might have has no meaning in the context of rigid body motion.

"*Motion*" stands for displacement, velocity and acceleration. Displacement analysis looks at the properties of the rigid body "mapping" from an initial pose to a final pose, without taking into account how the body has actually made the move. Velocity analysis looks at the *local* (also called *first order*, or *instantaneous*) properties of these transformations, i.e., when final and initial positions are "infinitesimally" separated, both in time and in space. Acceleration analysis looks at the *second-order* properties of the motion.

Velocity and acceleration. Everybody knows what the velocity and acceleration of a *point* in E^3 mean: they are the three-vectors that represent the first and second-order time derivatives of the point's position three-vector. But what are the velocity and the acceleration of a *rigid body*? Are they the time derivatives of any position representation? The answer to this seemingly trivial question is "No"! (See the following Chapters for the details.) So, we *define* the velocity and acceleration of a rigid body as follows:

Definition 1 (Rigid body velocity and acceleration) *The velocity and acceleration of a moving rigid body are given by any set of parameters that allow to find the velocity and acceleration of <u>any</u> point moving together with the rigid body.*

This definition probably only becomes clear in the following Chapters. For the time being, just remember that the motion of a moving rigid body is not a trivial extension of a moving point.

3.3 SE(3): the Lie group of displacements

This Section deals with (finite) *displacements* of rigid bodies. Imagine a rigid body B somewhere in the space around you. For the sake of simplicity, assume B is a cube. Move it from its current pose L to any other pose K you choose; then move it further to still another pose J. Observe that

1. The motion of B from L to K is *continuous*. That means that it could be broken up in arbitrarily small sub-motions.
2. Moving B from L to K, and then further to J, gives the same result as moving it from L "directly" to J ("*transitivity*")

Now translate B over a certain distance in the direction of one of its own edges (let's call this edge $\boldsymbol{e}_1$). Then rotate B over a certain angle about one of its *other* edges (let's call this edge $\boldsymbol{e}_2$). Start all over again, but change the sequence of motions: first rotate about $\boldsymbol{e}_2$, then translate along $\boldsymbol{e}_1$. Observe that

3. Rotations and translations do *not commute*.

Some properties you can test only in your mind, but nobody will contest that

4. All possible poses of B can be reached from any given initial pose.

5. All possible poses of B can be reached by combining translations along, and rotations about, three non-parallel edges of the object. One says that a rigid body has six *degrees of freedom*.
6. No initial pose has any special properties provided that the ambient space is completely empty. In other words, there is no natural "*origin*."

These coordinate-independent properties correspond to the mathematical concept of a *Lie group*, [21, 28, 33]. These continuous groups were first studied in great detail by the Norwegian mathematician Sophus Lie (1842–1899), [22], in his research on differential equations. The Lie group of rigid body displacements is called SE(3): the *Special Euclidean group in three dimensions*. It represents orientation- and distance-preserving transformations in E^3. "Distance-preserving" means that a displacement of a rigid body does not change the distances between any two of its points. "Orientation-preserving" means that a right-handed reference frame on a rigid body remains a right-handed reference frame, irrespective of what displacement is applied to the rigid body. This is obvious for rigid body displacements, but transformations exist in E^3 that do change the handedness of any reference frame without violating the distance constraints; for example, mirroring through a plane. The keyword "Special" distinguishes between both cases. Note the following important differences between E^3 (the "space of points") and SE(3) (the "space of frames") :

Fact-to-Remember 5 ($\mathbf{E}^3$ vs. SE(3))
E^3 is a three-dimensional space, while SE(3) is six-dimensional. A point in SE(3) corresponds to a frame in E^3.

Algebraic properties. This paragraph transforms the qualitative discussion of the previous paragraph into a formal, algebraic description. A Lie group has two basic properties: it's an algebraic *group*, and its group operation is *continuous*, [34, 35, 8, 17, 21]. Hence, the composition of elements in a Lie group has the following properties:

- The composition of a displacement g with a a displacement h is again a a displacement:
$$\forall g, h \in \mathrm{SE}(3) : g \circ h \in \mathrm{SE}(3), \text{and } h \circ g \in \mathrm{SE}(3). \tag{3.1}$$
We often use the shorthand "multiplicative" notation gh to denote the group composition operation $g \circ h$, i.e., first execute h and then g.
- The composition of displacements is a *continuous* operation, i.e., it can be subdivided in arbitrary small components, that each are displacements themselves.
- The composition of displacements is *associative*:
$$\forall g, h, l \in \mathrm{SE}(3) : (g \circ h) \circ l = g \circ (h \circ l). \tag{3.2}$$
- Not moving at all is also a displacement, called the *identity element* e, or the *neutral element*, in the group.
- Each displacement has an *inverse*:
$$\forall g \in \mathrm{SE}(3), \exists h \in \mathrm{SE}(3) : g \circ h = e = h \circ g. \tag{3.3}$$
This inverse is denoted by the classical multiplicative inverse: $h = g^{-1}$.
- The composition of displacements is, in general, *not commutative*:
$$\exists g, h \in \mathrm{SE}(3) : gh \neq hg. \tag{3.4}$$
Note that some displacements do commute; for example: two translations; or two rotations about the same axis: or the composition of any displacement with the identity or with its inverse displacement.

The manifold of rigid body motions. Besides being a continuous group, the space of rigid body displacements has another important structural property: SE(3) is a *manifold*. This means that *locally* it looks like the six-dimensional Euclidean space $\mathbb{R}^6$: one can use six real numbers to *represent* displacements, and these coordinate representations are smooth, i.e., their derivatives of all orders are continuous. The emphasis above is on the word "locally." Indeed, it is not difficult to see that SE(3) is *not globally* identical to $\mathbb{R}^6$:

1. A rigid body moving with a constant pure angular velocity will return to its original pose; applying a constant velocity to a point in $\mathbb{R}^6$ will never bring it back to where it started.
2. $\mathbb{R}^6$ is a *vector space*; SE(3) is not. In a vector space, the following property holds: $\lambda\mathcal{O}(a,b) = \mathcal{O}(\lambda a, \lambda b)$, where $\mathcal{O}$ denotes the composition operation in the space. Recall that the composition of displacements is a "multiplicative" operation, hence "λg" ($\lambda \in \mathbb{R}, g \in \mathrm{SE}(3)$) in the sense of applying g a number λ of times, is actually $g^\lambda = g\, g \dots g$. Hence, a counterexample proving that SE(3) is not a vector space is easily constructed: imagine moving the rigid body along a helical staircase, one floor up (this is the motion "h"). Then turn it upside down (this is the motion "g"); don't forget to turn the "upward" direction of the staircase too, i.e., this direction is connected to the moving body, and not to the world. Take $\lambda = 2$. Then $\lambda\mathcal{O}(g,h) = (gh)(gh)$ brings the body back to its initial pose, while $\mathcal{O}(\lambda g, \lambda h) = gghh$ brings it to the second floor, in upright position.

In fact, SE(3) is an example of a *curved space*. This implies that in order to properly describe SE(3) one needs exactly the same tools from differential geometry as needed for the description of the curved space-time of Einstein's general relativity! This book will not take this route, since this "omission" will not compromise the validity of the presented material. If, however, after digesting this book you have become interested in more advanced robotics topics, (such as nonlinear control, higher-order robot kinematics, optimal and non-holonomic motion planning, etc.), you will certainly benefit from a bit more differential geometry, starting with textbooks that are quite accessible to engineers, e.g., [5, 20, 26, 29, 34, 35, 36]. Anyway, you should at least get the following message from this Section:

Fact-to-Remember 6 (SE(3) is a curved space)
And hence some of the familiar properties of flat Euclidean space do not hold: orthogonality, straight line, etc.

3.4 se(3): the Lie algebra of velocities

This Section treats *velocities* in a coordinate-independent way. The literature often uses alternative terms for velocity analysis, such as *instantaneous kinematics*, or *first-order kinematics*, especially if also the second-order (or acceleration) analysis of the body's motion is discussed.

Tangent space at the identity. Imagine a rigid body at rest in a given position and orientation in space. (Remember that the pose of a rigid body is a *point* on the manifold SE(3).) Now attach reference frames to all points of the rigid body, even to points that lie outside of the body but are thought to be rigidly connected to it. Different frames connected to the same point on the body correspond to different points in SE(3). So do any two frames connected to different points of the body. Now pick one of these frames. Without loss of generality, this frame can be chosen as the *identity element* "e" of SE(3). When moving the body, the picked frame moves together with the body. Call $g(t)$ the frame's motion over time; this $g(t)$ is also a curve on the manifold SE(3). Assume that at time $t = 0$, the body is at the identity element $e = g(0)$. Then $\partial g/\partial t|_{t=0}$ is the *tangent vector* to the curve $g(t)$ at the identity element of the manifold. Similarly, one can trace all possible motions of the body through the identity element and construct their tangent vectors. All these tangent vectors span the *tangent space* to the manifold at the identity. Mathematicians have given this tangent space at the identity element a special name: "se(3)," [34, 35]. In other words, se(3) is the space of all possible velocities of that particular reference frame on the rigid body that instantaneously coincides with the chosen world reference frame. Note that *any* frame on the rigid body could have served as world reference frame, or, equivalenty, as identity element of SE(3).

Tangent bundle. The above-mentioned process of constructing tangent spaces is exactly similar to constructing the tangent space to a curved surface in E^3. The space of rigid body motions, however, is six-dimensional, and not three-dimensional like E^3. What was done for the particular frame chosen above can be repeated for any other frame on the moving body: define the local tangent space as the set of tangent vectors to all possible curves through the frame. This set of all tangent spaces at all points is called the *tangent bundle*.

Identification of tangent spaces. Since all frames fixed to a moving rigid body do not move with respect to each other, the time derivative of a frame that is not instantaneously at the identity contains the same information as the time derivative of that frame that is instantaneously at the identity. Since SE(3) is not a flat Euclidean space, Fact 6, it is not obvious how to compare and/or add tangent vectors at different points on the manifold. For example, try to imagine how you would add a vector tangent to the earth surface at the north pole to a vector tangent to the earth surface at the south pole. This kind of operation requires a rule of *parallel transport* between the tangent spaces at both points, e.g., [5, 35]. Another name for the same concept is *identification* of both tangent spaces, i.e., a rule that says what tangent vector in the first space corresponds to a given tangent vector in the second space. For SE(3), as well as for the earth surface, there is no unique ("natural," "canonical") way to define such a rule. In any Lie group, however, at least one possible coordinate-independent rule of parallel transport is always defined: *left translation.* This works as follows. Take a frame attached to the moving rigid body, not at the identity displacement e, but at some arbitrary other element g of the manifold SE(3). This frame at g also describes a curve over the manifold due to the motion of the rigid body. Each point on this curve in the close vicinity of g can be mapped to a point in the close vicinity of the identity element e by pre-multiplication ("left translation") with g^{-1}. (In a Lie group, this inverse of g is always well defined.) Now, the tangent vector to this left-translated curve at the identity e is an element of se(3). Hence, the tangent vector at g has been uniquely identified with a tangent vector at e. "Right translation" is defined in a completely similar way. However, the left and right translations of the same tangent vector need not coincide.

Addition—Vector space. The following paragraphs compare the properties of rigid body displacements to those of rigid body velocities at the identity. The major differences are:

1. Velocities compose in an *additive* way to yield new velocities, while displacements follow a *multiplicative* composition rule.
2. The composition of velocities is a *commutative* operation.
3. There exists a natural *origin*: the zero velocity.

Hence, instantaneous velocities (at the identity element) of a moving rigid body have the algebraic properties of a *vector space*: every linear combination of velocities is again a velocity: $\lambda(v + w) = \lambda v + \lambda w$. Recall that this linearity property does not hold for finite displacements: $(gh)^\lambda \neq g^\lambda h^\lambda$.

Multiplication—Lie bracket. se(3) has even more algebraic structure than just that of a vector space: it is a so-called *algebra.* An algebra is a space with *two* operations, [9] [16, p. 278]: "addition" and "multiplication." The space of all $n \times n$ matrices with addition and matrix multiplication is a well-known example. Multiplication of rigid body velocities is much less intuitive than the above-mentioned addition, but the following example illustrates the concept. Give the rigid body B a velocity in a certain direction $\boldsymbol{e}_1$ (this direction is determined with respect to the body itself), and move it during a short period of time; call this motion g. Then give it a velocity in a different direction $\boldsymbol{e}_2$, and move it again during a short period; call this motion h. The third motion is the inverse of the first one: move with the inverse velocity along $\boldsymbol{e}_1$. (Note that $\boldsymbol{e}_1$ has not changed with respect to the body, but it has changed with respect to any world reference!). Finally, execute the inverse of the motion in the direction $\boldsymbol{e}_2$. Figure 3.1 sketches this four-motion operation. This composition of four operations $h^{-1}g^{-1}hg$ is the *commutator* of the finite displacements g and h. In general, this commutator will not bring B back to its original position and orientation. Now imagine that the motions g and h tend to infinitesimally small motions, or, in other words, take the limit, for the time going to zero, of the commutator divided by the short time period during which the motions are executed. This means that the infinitesimal displacements g and h become tangent vectors to the trajectories, i.e., *velocities* v and w. Because of the limit process, these velocities apply at the "identity element," i.e., the undisplaced pose of the body B. Hence, the commutator above is called the *Lie derivative* or *Lie bracket*, denoted by $\mathcal{L}_v w$ or $[v, w]$, respectively. Hence, $[v, w]$ is a mapping from two tangent vectors at the identity element to a third vector at the identity element: $[\cdot, \cdot] : \mathrm{se}(3) \times \mathrm{se}(3) \to \mathrm{se}(3) : v, w \mapsto [v, w]$. It has the physical units of an *acceleration.* Any rule of parallel transport also transports the definition of the Lie bracket from the tangent space at the identity to the tangent space at any other element g of SE(3): first transport the two tangent vectors v and w to the identity; then apply the Lie bracket to these two transported tangent vectors; bring the resulting tangent vector at the identity back to the original tangent space; and *define* this last vector to be the Lie bracket of v and w. Note that the Lie bracket in the tangent space at the identity element of a Lie group is an intrinsic feature of that Lie group; the Lie bracket at an arbitrary other element of the Lie group depends on a rule of parallel transport. For example, left and right translation define *different* Lie brackets.

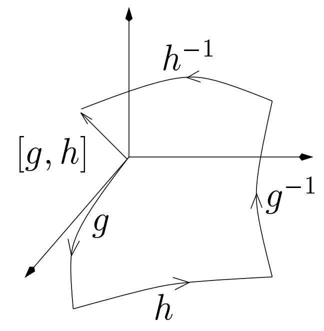


Figure 3.1: Commutator of two infinitesimal displacements.

Lie algebra. se(3) is not only a vector space under addition of velocities. It is also *closed* under the Lie bracket operation (*if* one considers velocities and their brackets to lie in the same space!), *and* the Lie bracket operation is distributive with respect to the addition of velocities: $[v, w + x] = [v, w] + [v, x]$. However, the rigid body Lie bracket operation does *not* have a neutral element (i.e., a velocity that commutes with *all* other velocities). And hence, also inverse elements are not defined. All these properties make se(3) into an algebra. Moreover, also the following properties always hold, [34]:

1. The Lie bracket is *continuous.*
2. The Lie bracket is *anti-symmetric*:
$$[v, w] = -[w, v]. \tag{3.5}$$
3. The Lie bracket satisfies the *Jacobi identity*, [9]:
$$[v, [w, x]] + [w, [x, v]] + [x, [v, w]] = 0. \tag{3.6}$$

The physical meaning of this Jacobi identity is not really obvious. Anyway, this property is not used in this introductory book.

Hence, the algebra se(3) is a *Lie algebra*. The name of course reflects the close relationship with Lie groups; see Section 3.5. History has chosen the adjective "Lie" instead of "Killing," although the German mathematician Wilhelm Karl Joseph Killing (1847–1923) introduced the Lie algebra concept independently of Lie in his study of non-Euclidean geometry. The notations "SE(3)" for the Lie group of displacements and "se(3)" for the Lie algebra of velocities at the identity respect the common practice of denoting Lie groups with capital letters and their associated Lie algebras with small letters.

Fact-to-Remember 7 (SE(3) vs. se(3))

SE(3) is the group of finite displacements. Finite displacements compose multiplicatively, and do, in general, not commute. se(3) is the vector space of velocities at the identity displacement. These velocities can be added, which is a commutative operation. This vector space is also endowed with the Lie bracket as non-commutative multiplication.

Distance measure. It is impossible to come up with a natural definition for the "length" of a rigid body motion in SE(3), i.e., one which has the same universal validity as the Euclidean distance function (1.5) in E^3, [21, 24]. The problem is that there is no prescribed way to "weight" the contributions of translation and rotation. Similarly, no natural "distance" between two elements of se(3) exists. This implies that many results derived on the basis of distance functions depend on the chosen weight. One important example in robotics is *data fitting*: one has measured a large sample of rigid body positions and orientations, and one now wants to find the "best" fitting pose. This is usually performed by a *least-squares* method that finds the body pose whose "distance" to all measurements is smallest. This result changes with a change in weight function in the distance calculations.

Fact-to-Remember 8 (Distance measure on SE(3) or se(3))
Neither SE(3) nor se(3) have a natural distance function ("metric").

3.5 Exp and log: from se(3) to SE(3) and back

Obviously, the algebra se(3) of rigid body velocities at the identity and the group SE(3) of rigid body displacements are closely related. One possible way to see this is by giving the body a certain *constant velocity* during *one unit of time*. At the end of this time interval, the body is at a certain position and orientation. A different initial velocity results in a different pose, and each pose can be reached by some velocity. (Actually, these mappings are not globally one-to-one: for example, rotation of the body over an angle of α degrees or an angle of $\alpha + n\times$ 360 degrees. We will not consider these "multiple coverings.") The mapping from a velocity to a pose is called the *exponentiation* or *exponential* of the velocity: exp : se(3) $\rightarrow$ SE(3). The inverse mapping from pose to velocity is called the *logarithm* of the pose: log : SE(3) $\rightarrow$ se(3). It maps the pose to a velocity that generates this pose in one unit of time. The names "exponential" and "logarithm" are no coincidence: as proven in a later Chapter, these mapping correspond exactly to the familiar exponentiation and logarithm for matrix representations of poses and velocities.

3.6 SO(3) and so(3)

A general motion of a rigid body has translational as well as rotational components. The special subclass of *motions with one fixed point* are the rigid body *rotations*. The rotational "displacements" form the three-dimensional subgroup SO(3) of SE(3). The "O" stands for "*Orthogonal*," and originates from the fact that rotations can be represented by orthogonal matrices, Chap. 5.

SO(3) is also a Lie group, and its corresponding Lie algebra of angular velocities is denoted by so(3). The Lie bracket on so(3) corresponds to the classical vector product.

The geometric and algebraic particularities of SE(3) are mainly consequences of the properties of the rotations; translations of a rigid body are completely equivalent to the motions of a point in E^3, with the simple vector space $\mathbb{R}^3$ as a faithful mathematical representation. Combining translations and rotations has one important property: translations and rotations do not commute in general. Hence:

Fact-to-Remember 9 (SE(3) is not SO(3) $\times\mathbb{R}^3$)
General rigid body motions cannot be decoupled into $\underline{\textit{independent}}$ *rotation and translation components: composing the rotation and translation components of two rigid body motions separately in SO(3) and $\mathbb{R}^3$, respectively, and then combining the results does not give the same motion as when the composition of the two motions is done in SE(3).*

3.7 SE(2) and se(2)

Many practical problems do not require the full six-dimensional arena offered by SE(3), but are mainly "planar" tasks. For example, moving a mobile robot over a factory floor; sorting packages on a conveyor belt; programming many spray-painting jobs; laying bricks; assembling printed circuit boards, etc. Basically, these jobs rely on displacements of *a frame in a plane*. These displacements have two translational degrees of freedom and one rotational degree of freedom. The two corresponding algebraic spaces are the Lie group SE(2), (i.e., the Special Euclidean group in two dimensional Euclidean space), and its Lie algebra se(2). Both are three-dimensional spaces, and they inherit the fundamental structure of their six-dimensional cousins: translational and rotational displacements do not commute, and no natural distance measure exists.

3.8 Velocity and acceleration vector fields

Any continuous sequence of poses that a rigid body travels through during a given time interval also determines its velocity and acceleration at each instant in that interval. "Velocity" and "acceleration" are well-known concepts for moving *points* in E^3; this Section takes a closer look at what exactly they mean for moving *rigid bodies*.

Velocity vector field. Choosing one tangent vector at each point of the manifold is called a *velocity vector field*. Any moving body generates such a vector field: at each point on the manifold (i.e., at each frame connected to the moving body) one attaches the tangent vector that corresponds to the derivative of the motion followed by that point (i.e, the velocity of the frame if it were rigidly connected to the moving body). Of course, it is not hard to imagine that not every vector field corresponds to a physically feasible motion of a rigid body. The tangent vectors are tangent to the six-dimensional manifold SE(3) of rigid body poses, and six-dimensional spaces are rather tough on the imagination. However, there exists a more intuitive representation in E^3: each point P in E^3 carries *two* three-vectors, one for the linear velocity of the point in the moving body that instantaneously coincides with P (i.e., the origin of the above-mentioned moving frame), and a second three-vector that represents the angular velocity of that frame. Some important properties of the velocity vector field of a moving rigid body are that

1. The velocity vector field *depends linearly* on the velocity: move the body twice as fast and the vectors in each point will be twice as large.
2. Given the velocity vector field at each instant in time, one can *reconstruct* the motion of the moving body.

Acceleration vector field. Similarly to the "twin" velocity vector fields on E^3 described in the previous paragraphs, one can attach *four* three-vectors to each point P in E^3 to represent the rigid body's acceleration: the first two give the linear velocity and acceleration of the origin of the rigid body frame that coincides instantaneously with P, and the third and fourth three-vectors represent the frame's angular velocity and acceleration. Note that (i) the knowledge of the two acceleration vectors at each instant in time is not sufficient to reconstruct the body's motion since also the velocity at the given time instant has to be known, and (ii) the acceleration vector field does not scale linearly with the velocity.

3.9 Twists and wrenches

Twists. Section 3.8 introduced the "twin" velocity vector field on E^3. Such a field has infinitely many vectors at each instant in time, which is not a very practical or economical way to describe a motion. However, since the moving body is rigid, all tangent vectors in the velocity vector field can be deduced from *any single one* of them. Hence, such a single tangent vector suits our Definition 1 of "rigid body velocity." One particular choice of tangent vector was already used in the 18th century by the Italian mathematician Giulio Mozzi (1730–1813) [27] and in the 19th century by his German and English colleagues Julius Plücker (1801–1868) and Arthur Cayley (1821–1895), but it is currently best known under the a name given by Robert Stawell Ball in the 1870s, [2]:

Fact-to-Remember 10 (Twist)
If one has chosen a world reference frame in E^3 (which is equivalent to a choice of origin in SE(3)), then the element of the velocity vector field at the origin of this reference frame is called the twist of the moving body. The simplest way to look at a twist is as a couple of three-vectors: the first one represents the angular velocity of the moving body, the second one represents the linear velocity of the point on the body that instantaneously coincides with the origin of the world frame.

This definition of a twist is exactly equivalent to the definition of a tangent vector to the rigid body motion at the identity, Sect. 3.4. Hence, the space of twists (often referred to as the *twist space*) is just se(3).

In the mechanics literature, not only rigid body *velocities* are called twists, but *infinitesimal displacements* of rigid bodies are called twists too, and some papers even use twist to denote a *finite displacement* as well. Using the terms "velocity twist," "infinitesimal displacement twist," and "finite displacement twist," respectively, avoids these ambiguities. The reason for this confusion is that these motion concepts can all be represented by a vector of six numbers, Chapters 4 and 6, although the previous Sections have shown that their geometric and algebraic properties are very different.

Wrenches. Motion is important in robotics, but *forces* are too. The *statics* of a mechanical structure describes how forces working at different points of the structure are equivalent to one resultant force. "Force" in this context means the combination of a linear force three-vector and a moment of force three-vector. As in the case of the velocity vector field for a moving body, a *force "vector field"* can be defined on SE(3): the resultant force and moment on a body can be kept in equilibrium *at any point of the body* by a certain combination of a linear force and an angular momentum applied at that point. Hence, one could think of the whole space (i.e., SE(3) or E^3, according to what is regarded as the manifold) as filled with vectors at each point. On SE(3), the six-vector in this field that works at the origin is called the *wrench* that acts on the body; this terminology is also due to Ball, [2]. In the same way, E^3 is filled with *couples* of three-vectors at each point, one representing a linear force and the second one representing an angular moment.

Vector field vs. one-form The previous pargraphs introduced wrenches in the classical way, as couples of three-vectors in E^3. For most engineering purposes this way of presentation is sufficient, but for advanced topics such as nonlinear robot control, it is important to be aware of the following: twists are elements of the vector space se(3); wrenches cannot be elements of the same space, since they represent physically different things, namely forces. There is however a relationship between velocity and force that is physically unambiguously defined: a force working on a moving body generates *power*, or, stripping the time dimension from the velocity, a force working on an infinitesimally moved body generates *work*. Hence, a wrench send a twist (i.e., tangent vector) onto a real number (i.e., work, or power) as a *linear* mapping. We've seen the same concept in Sect. 2.3 (i.e., a probability *measure*) and called it a *differential form*. Any given wrench is only a one-dimensional vector space, and is hence called a *one-form*. The vector space of *all* possible wrenches is six-dimensional, and is called the *co-tangent space*, $\mathrm{se}^*(3)$. (The name "co-tangent" is a bit misleading, since a wrench is tangent to nothing...) The tangent space of twists is the *dual* space of the vector space of wrenches, and vice versa, [9, 10]. In summary:

Fact-to-Remember 11 (Twists are tangent vectors, wrenches are one-forms)
Twists are tangent vectors to mappings $\mathbb{R} \rightarrow SE(3)$, i.e., mappings of an instant of time to a pose of a moving rigid body. Wrenches are <u>one-forms</u> on se(3), i.e., linear maps se(3) $\rightarrow \mathbb{R}$ that map a twist to the power generated by a rigid body that moves with this twist against the given wrench. The vector space of wrenches is the <u>dual</u> of the vector space se(3) of twists, and is denoted by $se^*(3)$. These names are not so important; what is important is to realise that twists and wrenches are <u>different</u> things!

You might think that the above emphasis is a bit exaggerated. But be aware when you start reading the literature on, for example, force control of robots, or on motion planning, since these research areas contain abundant examples of papers in which this distinction between vectors and one-forms is not recognized, and hence in which erroneous, physically nonsensical conclusions are drawn. See e.g., [4, 11] for concise introductions to some of the cultivated mistakes.

Dual bases The power operation between a twist $\mathbf{t}$ and a wrench $\mathbf{w}$ is called the *pairing* of the tangent vector and the co-tangent vector, and denoted by $\langle \mathbf{t}, \mathbf{w} \rangle$. That pairing is called *natural*, because it is independent of any choice of reference frame or physical units. A special case of this relationship occurs when the power in the pairing vanishes:

Fact-to-Remember 12 (Reciprocal twist and wrench)
A twist t *and a wrench* w *are called* *reciprocal, [2, 12, 15, 23, 30], if*

$$\boxed{\langle \mathsf{t}, \mathsf{w} \rangle = 0.} \tag{3.7}$$

The natural pairing between tangent and co-tangent spaces induces a correspondence between tangent and co-tangent spaces; this correspondence is called an *identification.* In terms of twists and wrenches, this means that to each given twist t, there corresponds a wrench w, constructed as follows, [10, p. 61]: (i) choose a basis $\{\boldsymbol{t}_1, \dots, \boldsymbol{t}_6\}$ in the tangent space; (ii) the dual basis $\{\boldsymbol{w}_1, \dots, \boldsymbol{w}_6\}$ in the co-tangent space is uniquely defined by the constraints $\langle \boldsymbol{t}_i, \boldsymbol{w}_j \rangle = \delta_{ij}$, with $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ if $i \neq j$; (iii) choose a twist t with coordinates $(x_1, \dots, x_6) : \mathsf{t} = x_1\boldsymbol{t}_1 + \cdots + x_6\boldsymbol{t}_6$; (iv) the wrench w that has the same coordinates $(x_1, \dots, x_6)$ is then called *the* dual of the twist t. Note, however, that this identification $\mathsf{t} \leftrightarrow \mathsf{w}$ is not "natural," since it depends on the choice of basis $\{\boldsymbol{t}_1, \dots, \boldsymbol{t}_6\}$!

The combination of a pairing operation and an identification operation leads to a *metric* defined on the manifold, i.e., a way to measure the "length" of tangent vectors. Indeed, take a twist t; use the identification operation to find a corresponding wrench w; the pairing $\langle \mathsf{t}, \mathsf{w} \rangle$ is a real number that can be used as the square of the "norm" of t. Note however, that (i) the manifold of rigid body poses does not have a natural metric, [25], and (ii) the metric is not necessarily positive-definite (i.e, the "norm" can be zero or negative).

Instantaneous twist and wrench axes This paragraph gives, without proof, two important and intuitively clear theorems from the previous century whose application to the field of robotics is obvious. The first is from the French mathematician Michel Chasles (1793–1881), [7], and states that

Fact-to-Remember 13 (Chasles' Theorem, 1830)
The most general motion for a rigid body is a screw motion, [1, 2, 3, 14], i.e., there exists a line in space (called the "screw axis" (SA), [2, 19, 32], or "twist axis") such that the body's motion is a rotation about the SA plus a translation along it.

Chasles himself formulated his principle in many different ways. The one that comes closest to the modern formulation is probably the following, [7, p. 321]: *On peut toujours transporter un corps solide libre d'une position dans une autre position quelconque déterminée, par le mouvement continu d'une vis à laquelle ce corps serait fixé invariablement.* Note that the motions considered above are *finite displacements.* If one brings the final position and orientation of the body closer and closer to the initial position (or, equivalently, one considers the position and orientation of a *moving* body at two close instants in time) the screw axis is called the *instantaneous* screw axis (ISA) or *twist axis* of this velocity or infinitesimal displacement. The notion of twist axis was probably already discovered many years before Chasles (the earliest reference seems to be the Italian Giulio Mozzi (1763), [6, 13, 27]) but he normally gets the credit.

Wrenches also posses a screw axis. This was formulated by the French geometer Louis Poinsot (1777–1859) [31], in a theorem similar to Chasles':

Fact-to-Remember 14 (Poinsot's Theorem, 1804)
Any system of forces applied to a rigid body can be reduced to a single force and an angular moment in a plane perpendicular to the force.

3.10 Constrained rigid body

A free rigid body has six degrees of motion freedom, and can resist no forces. If constraints act on the body, its motion degrees decrease, and the space of forces it can resist increases in dimension. Several sorts of constraints exist:

- *Hard constraint* (or *geometric constraint*, or *holonomic constraint*). The space of possible twists becomes lower-dimensional, since the motion of the rigid body is constrained in certain directions by contact with another rigid body.
- *Stiffness constraint.* The body is contacting an elastic body (or suspended on elastic bars or strings). It still has six motion degrees of freedom, but can now resist an n-dimensional vector space of wrenches (with $n > 0$), i.e., those that generate a deformation of the elastic constraining bodies. Stiffness is a mapping from infinitesimal displacement twists into wrenches.
- *Damping constraint.* Similar to a stiffness constraint, but the physical interpretation of damping is a mapping from velocity twists into wrenches.
- *Inertia constraint.* Again similar to the stiffness and damping constraints. Inertia maps velocity twists into *momentum* of a rigid body, see Chap. 10.

Most often, only the *linear* parts of the stiffness, damping, and inertia mappings are used, which can hence be *represented* by the stiffness, damping, and inertia *matrices*, respectively. The inverses of these mappings are called *compliance*, *accommodation*, and *mobility*, respectively. All real-world objects have a specific stiffness, damping, and inertia. Together, these form the so-called *impedance* of the object; the inverse is the *admittance*, [18].

References for this Chapter

[1] J. Angeles. *Rational Kinematics.* Springer Verlag, 1988.

[2] R. S. Ball. *Theory of screws: a study in the dynamics of a rigid body.* Hodges, Foster and Co, Dublin, Ireland, 1876.

[3] A. Bottema and B. Roth. *Theoretical Kinematics.* Dover Books on Engineering. Dover Publications, Inc., Mineola, NY, 1990.

[4] H. Bruyninckx. Some invariance problems in robotics. Technical Report 91P04, Katholieke Universiteit Leuven, Dept. Mechanical Engineering, Belgium, 1991.

[5] W. L. Burke. *Applied differential geometry.* Cambridge University Press, 1992.

[6] M. Ceccarelli. Screw axis defined by Giulio Mozzi in 1763. In *9th World Congress IFToMM*, pages 3187–3190, Milano, Italy, 1995.

[7] M. Chasles. Note sur les propriétés générales du système de deux corps semblables entr'eux et placés d'une manière quelconque dans l'espace; et sur le déplacement fini ou infiniment petit d'un corps solide libre. *Bulletin des Sciences Mathématiques, Astronomiques, Physiques et Chimiques*, 14:321–326, 1830.

[8] D. P. Chevallier. Lie algebras, modules, dual quaternions and algebraic methods in kinematics. *Mechanism and Machine Theory*, 26(6):613–627, 1991.

[9] P. M. Cohn. *Algebra.* Wiley, Chichester, 2nd edition, 1991.

[10] C. T. J. Dodson and T. Poston. *Tensor geometry: the geometric viewpoint and its uses*, volume 130 of *Graduate Texts in Mathematics.* Springer-Verlag, 2nd edition, 1991.

[11] J. Duffy. The fallacy of modern hybrid control theory

that is based on "orthogonal complements" of twist and wrench spaces. *J. Robotic Systems*, 7(2):139–144, 1990.

[12] R. Featherstone. The calculation of robot dynamics using articulated-body inertias. *Int. J. Robotics Research*, 2(1):13–30, 1983.

[13] C. G. Gibson and K. H. Hunt. Geometry of screw systems—1. Screws: Genesis and geometry. *Mechanism and Machine Theory*, 25(1):1–10, 1990.

[14] H. Goldstein. *Classical mechanics.* Addison-Wesley Series in Physics. Addison-Wesley, Reading, MA, 2nd edition, 1980.

[15] W. R. Hamilton. *Elements of quaternions*, volume I-II. Chelsea Publishing Company, New York, NY, 3rd edition, 1969. (Originally published in 1899).

[16] I. N. Herstein. *Topics in Algebra.* John Wiley & Sons, 2nd edition, 1975.

[17] J. M. Hervé. The mathematical group structure of the set of displacements. *Mechanism and Machine Theory*, 29(1):73–81, 1994.

[18] N. Hogan. Impedance control: An approach to manipulation. Parts I-III. *Trans. ASME J. Dyn. Systems Meas. Control*, 107:1–24, 1985.

[19] K. H. Hunt. *Kinematic Geometry of Mechanisms.* Oxford Science Publications, Oxford, England, 2nd edition, 1990.

[20] A. Isidori. *Nonlinear Control Systems.* Communications and Control Engineering Series. Springer Verlag, Berlin, Germany, 2nd edition, 1989.

[21] A. Karger and J. Novak. *Space kinematics and Lie groups.* Gordon and Breach, New York, NY, 1985.

[22] S. Lie. *Theorie der Transformationsgruppen.* B. G. Teubner, Leipzig, Germany, 1888. Three volumes.

[23] H. Lipkin and J. Duffy. Hybrid twist and wrench control for a robotic manipulator. *Trans. ASME J. Mech. Transm. Automation Design*, 110:138–144, 1988.

[24] J. Lončarić. *Geometrical Analysis of Compliant Mechanisms in Robotics.* PhD thesis, Harvard University, Cambridge, MA, 1985.

[25] J. Lončarić. Normal forms of stiffness and compliance matrices. *IEEE J. Rob. Automation*, RA-3(6):567–572, 1987.

[26] C. W. Misner, K. S. Thorne, and J. A. Wheeler. *Gravitation.* W. H. Freeman, San Francisco, CA, 1973.

[27] G. Mozzi. *Discorso Matematico sopra il Rotamento Momentaneo dei Corpi.* Stamperia del Donato Campo, Napoli, 1763.

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

[29] H. Nijmeijer and A. van der Schaft. *Nonlinear Dynamical Control Systems.* Springer Verlag, New York, NY, 1996. Corrected 3rd printing.

[30] M. S. Ohwovoriole and B. Roth. An extension of screw theory. *Trans. ASME J. Mech. Design*, 103(4):725–735, 1981.

[31] L. Poinsot. Sur la composition des moments et la composition des aires. *Journal de l'Ecole Polytechnique*, 6(13):182–205, 1806.

[32] B. Roth. On the screw axis and other special lines associated with spatial displacements of a rigid body. *Trans. ASME J. Eng. Industry*, 89:102–110, 1967.

[33] A. E. Samuel, P. R. McAree, and K. H. Hunt. Unifying screw geometry and matrix transformations. *Int. J. Robotics Research*, 10(5):454–472, 1991.

[34] D. H. Sattinger and O. L. Weaver. *Lie groups and algebras with applications to physics, geometry, and mechanics*, volume 61 of *Applied Mathematical Sciences.* Springer-Verlag, New York, NY, 1986.

[35] B. F. Schutz. *Geometrical methods of mathematical physics.* Cambridge University Press, Cambridge, England, 1980.

[36] J. M. Selig. *Geometrical Methods in Robotics.* Springer Verlag, New York, NY, 1996.

Chapter 4

Screws: twists and wrenches

4.1 Introduction

After the previous Chapter's *coordinate-free* approach to modelling the *structure* of rigid body motion, this Chapter describes its *coordinate representations.* The basic geometric entities needed for rigid body motion representation are: points, vectors, lines, and screws.

Fact-to-Remember 15 (Basic ideas of this Chapter)
Most of classical mechanics deals with properties of <u>*points*</u> *or point masses. The study of rigid body mechanics, however, requires more:* <u>*lines*</u> *are important (e.g., to model the axes of the revolute or prismatic joints most robots are constructed with), and certainly* <u>*screws*</u>*, which generalise the line in the sense that both translational and angular components are described (cf. the Theorems of Chasles and Poinsot). So, it is important to know how lines and screws are represented in* <u>*coordinates*</u>*.*

4.2 Points

Points are the simplest geometric entities. A point's position $\boldsymbol{p}$ can be represented numerically by a coordinate three-vector $\boldsymbol{p} = (\boldsymbol{p}_x\, \boldsymbol{p}_y\, \boldsymbol{p}_z)^T$. Note the double use of the symbol $\boldsymbol{p}$: it denotes (i) the position of a point without referring to whatever reference frame, and (ii) the coordinates of the point with respect to a chosen *reference frame.* The distance between two points is given by the Euclidean distance function in Eq. (1.5), repeated here for convenience:

$$d(\boldsymbol{p}^1, \boldsymbol{p}^2) = (\boldsymbol{p}^1_x - \boldsymbol{p}^2_x)^2 + (\boldsymbol{p}^1_y - \boldsymbol{p}^2_y)^2 + (\boldsymbol{p}^1_z - \boldsymbol{p}^2_z)^2. \tag{4.1}$$

Point coordinates are often represented by a *homogeneous coordinates* four-vector, denoted by the same symbol: $\boldsymbol{p} = (\boldsymbol{p}_x\, \boldsymbol{p}_y\, \boldsymbol{p}_z\, 1)^T$. One of the reasons for this custom is that it allows to work with the points "at infinity" in the same way as the normal points; those points at infinity have a fourth component equal to 0.

4.3 Direction vectors

Vectors can be used to represent directions in the Euclidean space E^3. A frequently used representation is the unit sphere S^2 in E^3. (The superscript "2" refers to the two-dimensionality of the sphere's surface.) Each point of S^2 is the end point of a unique position vector starting at the origin of the sphere, and determines a spatial direction. Direction vectors have a *sense* too: they impose an ordering on all points on the line through the origin of S^2 and the chosen point on its surface. So, each line through the origin contains two direction vectors with opposite sense.

If you walk along a "straight line" over the unit sphere, you finally arrive at the point you started from; it would take you quite some time to repeat a similar experiment in the Euclidean space. . . The "straight lines" on S^2 are in fact the so-called *great circles*: the intersections of the sphere with planes through its origin. The distance between two directions (represented by two unit vectors $\boldsymbol{e}^1$ and $\boldsymbol{e}^2$) is also measured along these great circles: construct the (unique) great circle that contains the endponts of both direction vectors, and measure the distance along the unit sphere to travel from the end point of $\boldsymbol{e}^1$ to the end point of $\boldsymbol{e}^2$. In other words, this distance is the angle (in radians) between the two direction vectors:

$$d(\boldsymbol{e}^1, \boldsymbol{e}^2) = \arccos(\boldsymbol{e}^1 \cdot \boldsymbol{e}^2), \tag{4.2}$$

with the dot denoting the classical inner product between two Euclidean vectors. Note that the symbol $d(\cdot)$ has different meaning when working on points or on direction vectors.

Fact-to-Remember 16 (Geometry of S^2)
The "distance" between two directions is not represented by the Euclidean distance formula (1.5). The reason is that directions (i.e.,, the unit sphere S^2) have a different geometry than the points in the Euclidean space E^3, [17, 18, 19, 20].

Polar and axial vectors. Two interpretations exist for direction vectors in three-dimensional space, depending on how they incorporate the notion of *orientation*, [6, 11]:

1. *Axial* vectors have an *inner* orientation, i.e., the direction of the vector indicates the positive orientation. For example, a unit linear force vector: the positive direction of the force does not depend on the orientation (right-handed vs. left-handed) of the world reference frame.

2. *Polar* vector have an *outer* orientation, i.e., the positive orientation cannot be derived from the direction vector itself, but is imposed on it by the "environment." For example, a unit moment of force vector: if the handedness of the world frame changes, the orientation associated with the moment vector changes too. Note that this is a feature of the *coordinate representation*, not of the physical property that the vector stands for.

As many other textbooks, this book *implicitly* uses right-handed reference frames only, but no physical arguments prevent the use of left-handed frames.

Vectors are usually given three-vectors as coordinates, just as points. However, there is a fundamental difference between vectors and points: the addition of vectors is frame-independent, the "addition" of points not. For example: adding two velocity vectors of the same moving point mass has physical meaning; adding two positions has no such meaning. Hence, vectors are sometimes given homogeneous coordinate four-vectors with a zero fourth component: adding two such four-vectors gives another four-vector with zero fourth component. For points with a "1" fourth component, this addition does not work out.

4.4 Lines–Planes

Lines are very important in robotics because:

- They model *joint axes*: a revolute joint makes any connected rigid body rotate about the line of its axis; a prismatic joint makes the connected rigid body translate along its axis line.
- They model *edges* of the polyhedral objects used in many task planners or sensor processing modules.
- They are needed for *shortest distance* calculation between robots and obstacles.

Not only lines, but also *planes* are very common in the models used in robotics: in any simple environment model, planes will be used to represent obstacles and object surfaces. However, no new mathematical concept is required to model planes: the position of a point in the plane together with the (directed) normal line to the plane contain exactly the same information. Classical geometry defines a line in two fundamental ways: (i) as the "join" of two points, and (ii) as the "intersection" of two planes. However, a faithful representation of a joint needs one more piece of information on top of the geometric description for a line: the positive sense of the joint's motion.

4.4.1 Non-minimal vector coordinates

A line $\mathcal{L}(\boldsymbol{p}, \boldsymbol{d})$ is completely defined by the ordered set of two vectors, (i) one point vector $\boldsymbol{p}$, indicating the position of an arbitrary point on $\mathcal{L}$, and (ii) one free direction vector $\boldsymbol{d}$, giving the line a direction as well as a sense. (Note that the *sense* of the line is in fact not important if one just wants to represent an *undirected line.*) Each point $\boldsymbol{x}$ on the line is given a parameter value t that satisfies $\boldsymbol{x} = \boldsymbol{p} + t\boldsymbol{d}$. The parameter t is unique once $\boldsymbol{p}$ and $\boldsymbol{d}$ are chosen. The representation $\mathcal{L}(\boldsymbol{p}, \boldsymbol{d})$ is not *minimal*, because it uses six parameters for only four degrees of freedom. The following two constraints apply:

1. The direction vector can be chosen to be a *unit vector*, i.e., $\boldsymbol{d} \cdot \boldsymbol{d} = 1$.
2. The point vector $\boldsymbol{p}$ can be chosen to be the point on the line that is nearest the origin, i.e. $\boldsymbol{p}$ is orthogonal to the direction vector $\boldsymbol{d}$: $\boldsymbol{p} \cdot \boldsymbol{d} = 0$.

With respect to a world reference frame, the line's coordinates are given by a six-vector:

$$\mathbf{l} = \begin{pmatrix} \boldsymbol{p} \\ \boldsymbol{d} \end{pmatrix}. \tag{4.3}$$

4.4.2 Plücker coordinates

The previous section uses a point vector $\boldsymbol{p}$ and a free vector $\boldsymbol{d}$ to represent the line $\mathcal{L}(\boldsymbol{p}, \boldsymbol{d})$. Arthur Cayley (1821–1895) and Julius Plücker (1801–1861) introduced an alternative representation using *two free vectors*, [4, 7, 16, 22, 21]. This representation was finally named after Plücker. We denote this Plücker representation by $\mathcal{L}_{\mathrm{pl}}(\boldsymbol{d}, \boldsymbol{m})$. Both $\boldsymbol{d}$ and $\boldsymbol{m}$ are *free* vectors: $\boldsymbol{d}$ has the same meaning as before (it represents the direction of the line) and $\boldsymbol{m}$ is the *moment* of $\boldsymbol{d}$ about the chosen reference origin, $\boldsymbol{m} = \boldsymbol{p} \times \boldsymbol{d}$. (Note that $\boldsymbol{m}$ is independent of which point $\boldsymbol{p}$ on the line is chosen: $\boldsymbol{p} \times \boldsymbol{d} = (\boldsymbol{p} + t\boldsymbol{d}) \times \boldsymbol{d}$.)

The advantage of the Plücker coordinates is that they are homogeneous: $\mathcal{L}_{\mathrm{pl}}(k\boldsymbol{d}, k\boldsymbol{m}), k \in \mathbb{R}$, represents the same line, while $\mathcal{L}(k\boldsymbol{p}, k\boldsymbol{d})$ does not. (It will also extend in a natural way the representations of rigid body velocity, and of force and torque, Sect. 4.5.) A coordinate representation of the line in Plücker coordinates is the following six-vector $\mathbf{l}$:

$$\mathbf{l} = \begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{m} \end{pmatrix}, \tag{4.4}$$

with $\boldsymbol{d}$ and $\boldsymbol{m}$ the column three-vectors representing the coordinates of the direction and moment vectors, respectively. The two three-vectors $\boldsymbol{d}$ and $\boldsymbol{m}$ are always *orthogonal*:

$$\boldsymbol{d} \cdot \boldsymbol{m} = 0. \tag{4.5}$$

A line in Plücker representation has still only four independent parameters, so it is not a minimal representation. The two constraints on the six Plücker coordinates are (i) the homogeneity constraint (i.e., multiplying by a scalar k does not change the line), and (ii) the orthogonality constraint (4.5).

Fact-to-Remember 17 (Plücker coordinates)
The Plücker coordinates of a line consist of two three-vectors: (i) an <u>axial</u> direction three-vector, and (ii) the <u>polar</u> moment three-vector of this direction vector about the origin of the <u>chosen</u> reference frame. Only four of the six coordinates are independent.

Finding a point on the line. It is sometimes necessary to find a point on the line, when only its Plücker representation $\mathcal{L}_{\text{pl}}(\boldsymbol{d}, \boldsymbol{m})$ is known. The following reasoning leads to the point $\boldsymbol{p}$ closest to the origin of the reference frame: $\boldsymbol{d} \times \boldsymbol{m} = \boldsymbol{d} \times (\boldsymbol{p} \times \boldsymbol{d}) = \boldsymbol{p}(\boldsymbol{d} \cdot \boldsymbol{d}) - \boldsymbol{d}(\boldsymbol{d} \cdot \boldsymbol{p}) = \boldsymbol{p}(\boldsymbol{d} \cdot \boldsymbol{d})$, since $\boldsymbol{p}$ is normal to the line, i.e., $\boldsymbol{d} \cdot \boldsymbol{p} = 0$. Hence,

$$\boxed{\boldsymbol{p} = \frac{\boldsymbol{d} \times \boldsymbol{m}}{\boldsymbol{d} \cdot \boldsymbol{d}}.} \tag{4.6}$$

Intersection of lines. Plücker coordinates allow a simple test to see whether two (non-parallel) lines l^1 and l^2 intersect. This test is often applied to robot joint axes. The lines intersect, if and only if, (Fig. 4.1),

$$\boxed{\boldsymbol{d}^1 \cdot \boldsymbol{m}^2 + \boldsymbol{d}^2 \cdot \boldsymbol{m}^1 = 0.} \tag{4.7}$$

Proof: the lines intersect if the two lines are coplanar, i.e., the vector $\boldsymbol{r}^2 - \boldsymbol{r}^1$ from a point $\boldsymbol{r}^1$ on l^1 to a point $\boldsymbol{r}^2$ on l^2 lies in this plane. In other words, it is orthogonal to the common normal direction, given by $\boldsymbol{d}^1 \times \boldsymbol{d}^2$: $0 = (\boldsymbol{d}^1 \times \boldsymbol{d}^2) \cdot (\boldsymbol{r}^2 - \boldsymbol{r}^1) = (\boldsymbol{d}^1 \times \boldsymbol{d}^2) \cdot \boldsymbol{r}^2 - (\boldsymbol{d}^1 \times \boldsymbol{d}^2) \cdot \boldsymbol{r}^1 = -\boldsymbol{d}^1 \cdot (\boldsymbol{d}^2 \times \boldsymbol{r}^2) - (\boldsymbol{r}^1 \times \boldsymbol{d}^1) \cdot \boldsymbol{d}^2 = -\boldsymbol{d}^1 \cdot \boldsymbol{m}^2 - \boldsymbol{d}^2 \cdot \boldsymbol{m}^1$.

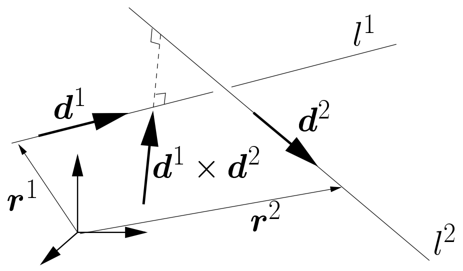


Figure 4.1: Two crossing lines l^1 and l^2, with common normal direction $\boldsymbol{d}^1 \times \boldsymbol{d}^2$. $\boldsymbol{r}^1$ and $\boldsymbol{r}^2$ denote points on the lines, with respect to the origin of the reference frame.

Common normal. The common normal line $\mathbf{l}^{cn}$ to two lines can be calculated as follows. Figure 4.1 shows that the following vector closure relation holds:

$$\boldsymbol{r}^1 + k\boldsymbol{d}^1 + l\boldsymbol{d}^1 \times \boldsymbol{d}^2 + m\boldsymbol{d}^2 = \boldsymbol{r}^2. \tag{4.8}$$

The scalars k, l, and m are still to be determined; $\boldsymbol{d}^1 \times \boldsymbol{d}^2$ is the direction vector of the common normal. Equation (4.8) can be written as a set of three linear equations in the three unknowns k, l, and m:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{4.9}$$

with $\boldsymbol{A} = \big(\boldsymbol{d}^1 \ (\boldsymbol{d}^1 \times \boldsymbol{d}^2) \ \boldsymbol{d}^2\big), \boldsymbol{x} = (k \ l \ m)^T$, and $\boldsymbol{b} = \boldsymbol{r}^2 - \boldsymbol{r}^1$. This linear set of equations can be solved for the unknowns k, l, and m. Hence, the Plücker coordinates of the common normal $\mathbf{l}^{cn}$ are

$$\mathbf{l}^{cn} = \begin{pmatrix} \boldsymbol{d}^1 \times \boldsymbol{d}^2 \\ \boldsymbol{p} \times (\boldsymbol{d}^1 \times \boldsymbol{d}^2) \end{pmatrix}, \quad \text{with} \quad \boldsymbol{p} = \boldsymbol{r}^1 + k\boldsymbol{d}^1. \tag{4.10}$$

4.4.3 Denavit-Hartenberg line coordinates

In the early 1950s, Jacques Denavit and Richard S. Hartenberg presented the first minimal representation for a line which is now widely used, (Fig. 4.2), [9, 14]. A line representation is *minimal* if it uses only *four* parameters, which is the minimum needed to represent all possible lines in E^3. The *common normal* between two lines was the main geometric concept that allowed Denavit and Hartenberg to find a minimal representation. The line $\mathcal{L}$ must first be given a direction, and is then described uniquely by the following four parameters:

1. The *distance* d: the orthogonal distance (i.e., along the common normal) between the line $\mathcal{L}$ and the line along the Z-axis of the world reference frame. The common normal's positive direction is from the Z-axis to the line; d is always a positive real number.

2. The *azimuth* α: the angle from the X-axis of the world reference frame to the projection of the common normal on the XY-plane. The positive sense of α follows the right-hand rule of rotations about the Z-axis of the world frame.

3. The *twist* θ: the rotation about the common normal that brings $\mathcal{L}$ parallel to the Z axis of the world frame. (Also the positive sense of both lines must match!) The sign of θ is determined by the right-hand rule of rotation about the (oriented) common normal.

4. The *height* h: the signed distance from the XY plane to the point where the common normal intersects the Z axis of the world reference frame. This Z-axis defines the sign of h.

The literature contains alternative formulations, differing mainly in the conventions for signs and reference axes. Conceptually, all these formulations are equivalent, and they represent the line $\mathcal{L}$ by two translational parameters (the distance d and the height h) and two rotational parameters (the azimuth α and the twist θ). We denote such a Denavit-Hartenberg representation ("DH representation," for short) as $\mathcal{L}_{dh}(d, h, \alpha, \theta)$. Note that a set of four DH parameters not only represents a (directed) *line*, but also the pose of a *frame*, that has its Z axis on the given line and its X axis along the common normal. Since only four parameters are used, the frames that can be represented this way satisfy two constraints: (i) their X-axis intersects the Z-axis of the world frame, and (ii) it is parallel to XY-plane of the world frame. An alternative interpretation of these constraints is that Z-axis of the frame can be freely chosen, but not the position of the frame's origin along the line, nor the orientation of the frame about the line.

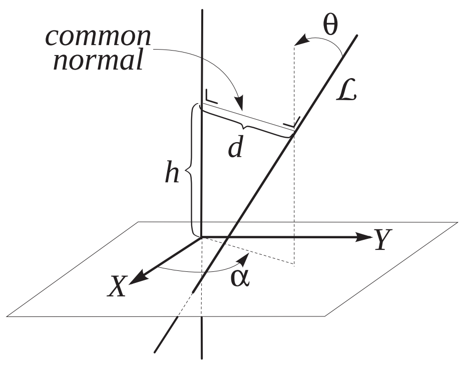


Figure 4.2: Line parameters in the Denavit-Hartenberg convention.

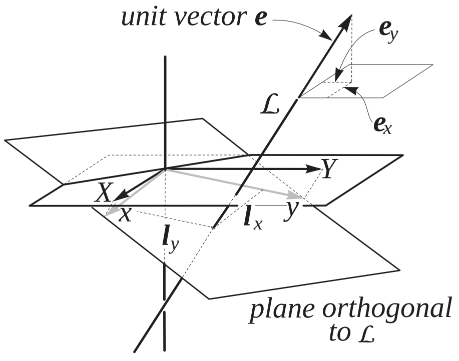


Figure 4.3: Line parameters in the Hayati-Roberts convention.

Fact-to-Remember 18 (DH representation and its singularities)
The DH representation is a minimal representation for a line with respect to a reference frame. It has problems to represent parallel lines, since then (i) the common normal is not uniquely defined, and (ii) the parameters change discontinuously when the line moves continuously through a configuration in which it is parallel to the Z axis. These two effects are examples of coordinate singularities.

The DH representation is *minimal*: it uses only four parameters to describe four degrees of freedom. Frames or lines do not form vector spaces (i.e., "adding" them is not a well-defined operation), and no representations exist that can represent them with a minimal number of paramaters *and* without singularities. This problem can be solved in two ways:

1. Using more than one so-called *coordinate patch*. For example: a complete map of the earth requires more than one sheet of paper. One has to know where the singularities in each patch are, to decide when to switch from one coordinate patch to the next.

2. Using more than four parameters for a line, or more than six for a frame, [24, 25, 30]. The price to pay with these non-minimal representations is that the parameters must always be kept consistent with a set of *constraints*.

Ambiguity. If someone gives you four numbers and tells you that they are DH parameters, you won't be able to know exactly what line or frame they represent: the interpretation of the four numbers requires a lot of *implicit* knowledge, such as the choice of right or left-handed reference frames, the origin of the inertial frame, its orientation, the positive directions of distances and angles. This problem can show up with any representation (minimal or not) but minimal representations suffer most.

Applications of line representations. The advantages and disadvantages of DH parameters play an important role in the *calibration* of a robot's geometrical model (i.e., the relative position and orientation of its links and joints). As with any man-made device, a robot's real geometry can differ from its nominal model, especially if the same model is used for all robots in mass production. Hence, if very accurate positioning is required, the nominal geometric model should first be calibrated for each device separately. That means that one determines the robot's geometric parameters from a set of accurately measured test motions. The most common approach to calibration is (i) to assume small errors in the nominal parameters, and (ii) to use some least-squares solution technique to derive the numerical values of these errors from the differences between the measurements and the predictions of the nominal model, [3]. A small number of parameters is an advantage for the numerical procedure; hence, minimal line representations are an advantage. On the other hand, commercial robots often have some axes that are assumed to be exactly parallel; hence, trying to find small parallelism errors with a representation that has a singularity exactly for parallel lines is very impractical.

Another important application for line representations is in the context of surface normal estimation: many sensor based robot tasks need accurate estimation of the normal to the surface that is being scanned (by a contact, distance, or force sensor, or looked at by a camera). All these estimation routines use some sort of coordinate line representation, and minimal and singularity-free representations have obvious advantages here too.

4.4.4 Hayati-Roberts coordinates

Another minimal line representation is the *Hayati-Roberts* line representation, that we denote by $\mathcal{L}_{hr}(\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{l}_x, \boldsymbol{l}_y)$, [3, 15, 23] (Fig. 4.3):

1. $\boldsymbol{e}_x$ and $\boldsymbol{e}_y$ are the X and Y components of a *unit* direction vector $\boldsymbol{e}$ on the line. The requirement that $\boldsymbol{e}$ be a unit vector eliminates the need for the Z component of the direction vector, since it is easily found as $\boldsymbol{e}_z = (1 - \boldsymbol{e}_x^2 - \boldsymbol{e}_y^2)^{1/2}$.
2. $\boldsymbol{l}_x$ and $\boldsymbol{l}_y$ are the coordinates of the intersection point of the line with the plane through the origin of the world reference frame, and normal to the line. The reference frame on this normal plane has the same origin as the world reference frame, and its X and Y frame axes are the images of the world frame's X and Y axes through parallel projection along the line.

This representation is unique for a directed line. Its coordinate singularities are different from the Denavit-Hartenberg singularities: it has no jumps in the parametrization if the line is (nearly) parallel to the world Z axis, but it does have singularities if the line becomes parallel to either the X or Y axis of the world frame.

4.5 Screws

Chasles' Theorem, Fact 13, says that with any instantaneous motion of a rigid body (a *twist*) there corresponds a line in space (the "screw axis"), on which two vectors $\boldsymbol{v}_{sa}$ and $\boldsymbol{\omega}_{sa}$ describe the body's translational and angular velocity, respectively. The body's velocity can also be represented with the same two vectors $\boldsymbol{d}$ and $\boldsymbol{m}$ of the Plücker line representation $\mathcal{L}_{\rm pl}(\boldsymbol{d}, \boldsymbol{m})$, (Sect. 4.4.2), by defining $\boldsymbol{d} = \boldsymbol{\omega}_{sa}$ and $\boldsymbol{m} = \boldsymbol{v}_{sa} + \boldsymbol{p}^{sa} \times \boldsymbol{\omega}_{sa}$, where $\boldsymbol{p}^{sa}$ is the position vector of a point on the screw axis. However, the two vectors $\boldsymbol{d}$ and $\boldsymbol{m}$ are then not necessarily orthogonal anymore, as was the case for a line. Such an "extended" line is called a *screw*, [1, 2, 4, 16, 29] and denoted by $\mathcal{L}_{\rm sc}(\boldsymbol{d}, \boldsymbol{m})$. A *wrench* is another physical instantiation of a screw: one vector is the linear force applied to the body along the line, and the second vector is the pure moment applied to the body about the line.

Pitch of a screw. The ratio between the parallel vectors $\boldsymbol{v}_{sa}$ and $\boldsymbol{\omega}_{sa}$ is called the *pitch* p of the screw:

$$\boxed{p = \frac{\boldsymbol{v}_{sa}}{\boldsymbol{\omega}_{sa}}.} \tag{4.11}$$

The names "screw" and "pitch" come from the similarity with the motion of a nut moving over a bolt; the amount of translation for each turn of the nut is indeed its pitch. Pitch has the physical dimensions of *length*.

Commutation of translation and rotation. Translational and rotational motions of a rigid body do not commute in general. However, the two motions described by the individual translational and rotational motion vectors *on the screw axis* do commute: first translating along the screw axis and then rotating about it results in the same motion as performing the translation and rotation in the opposite order. This is an example of a "property" of a *representation*, that is not a *structural* (or "*invariant*") property: representing the same motion in another frame makes the "property" disappear.

4.5.1 Screw coordinates

Once a world reference frame is chosen, the *coordinates* of the screw $\mathcal{L}_{\mathrm{sc}}(\boldsymbol{\omega}, \boldsymbol{v})$ form a six-vector $\mathbf{s}$:

$$\mathbf{s} \triangleq \begin{pmatrix} \boldsymbol{\omega} \\ \boldsymbol{c} \end{pmatrix} \triangleq \begin{pmatrix} \boldsymbol{\omega} \\ \boldsymbol{p} \times \boldsymbol{\omega} + \boldsymbol{v} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\omega} \\ \boldsymbol{p} \times \boldsymbol{\omega} + p\boldsymbol{\omega} \end{pmatrix}, \tag{4.12}$$

with $\boldsymbol{p}$ the (coordinate three-vector of the) position of *any* point on the screw axis, and p the pitch of the screw. Similarly to Eq. (4.6), it is straightforward to find the position vector $\boldsymbol{p}$ of the point on the line *closest to the origin*:

$$\boldsymbol{p} = \frac{\boldsymbol{\omega} \times \boldsymbol{c}}{\boldsymbol{\omega} \cdot \boldsymbol{\omega}}. \tag{4.13}$$

Hence, finding the screw axis from the screw coordinates is simple: $\boldsymbol{p}$ is a point on the screw axis, and $\boldsymbol{\omega}$ is a direction vector on the screw axis. Finding the pitch p is equally straightforward:

$$p = \frac{\boldsymbol{\omega} \cdot \boldsymbol{c}}{\boldsymbol{\omega} \cdot \boldsymbol{\omega}}. \tag{4.14}$$

A screw has *five* independent parameters: four for the line, plus one for the pitch. The only remaining constraint is the homogeneity constraint: the screws with six-vectors $\mathbf{s}$ and $k\mathbf{s}$ lie on the same line and have the same pitch.

4.5.2 Twist and wrench coordinates

Velocity twist. The homogeneity constraint represents the physical fact that a screw models *all possible* motions of a mechanical nut over a bolt with a given pitch. The constraint does not hold anymore if one uses a screw to represent a rigid body motion with a given translational or rotational *speed*. In these cases the *magnitudes* of $\boldsymbol{\omega}$ andr $\boldsymbol{c}$ are determined by this speed. Such a screw with a given speed is the *twist* of Sect. 3.9. (Sir William Rowan Hamilton (1788–1856) [13] called it a "screw with a magnitude." William Kingdon Clifford (1845–1879) called it a *motor*, [8, 26, 27, 28], this word being the contraction of "motion" and "vector.")

Fact-to-Remember 19 (Independent parameters for line, screw, twist)
A line in E^3 has $\underline{\text{four}}$ independent parameters, a screw has $\underline{\text{five}}$ independent parameters, and a twist has $\underline{\text{six}}$ independent parameters.

The screw coordinates $(\boldsymbol{\omega}^T\ \boldsymbol{c}^T)^T$ of the twist of a rigid body represent, respectively, the angular velocity of the body, and the translational velocity of the point on the rigid body that instantaneously coincides with the origin of the world frame, Fact 10.

Screw twist vs. pose twist. The robotics literature uses twist *coordinates* with two different interpretations:

1. References originating from screw theory (i.e., relying on basic sources such as [2, 16, 26] etc.) follow the just-mentioned interpretation: the last three-vector in the twist coordinates is the velocity of the *origin*.
2. Other references use this last three-vector in the twist coordinates to represent the velocity of a reference point on the moving body, *different* from the origin.

When needed, this book distinguishes between both interpretations by using the names *screw twist* and *pose twist*, respectively. (The motivation for the term "screw twist" should be clear by now; the motivation for "pose twist" is given in Chapter 6.) Pose twists should not be used in Eqs (4.13) and (4.14).

Infinitesimal displacement twist. A velocity twist could be interpreted as the limit case of the ratio of (i) an infinitesimal displacement of the rigid body, and (ii) the infinitesimal time interval during which the displacement takes place. The nominator of this ratio, i.e., the infinitesimal displacement, is also called an *instantaneous screw*, [2]. Its coordinates are

$$\mathbf{t}_\Delta = \begin{pmatrix} \delta_x \; \delta_y \; \delta_z \; d_x \; d_y \; d_z \end{pmatrix}^T . \tag{4.15}$$

$(\delta_x \, \delta_y \, \delta_z)^T$ represents small rotations about the X, Y and Z axes, respectively; $(d_x \, d_y \, d_z)^T$ is the three-vector of small translations along the axes. The same two interpretations as in the previous paragraph exist in the context of infinitesimal twists too.

Finite displacement. *Finite* displacements can also be represented by a screw-like six-vector $\mathbf{t}_d$: the direction of the first three-vector represents the orientation of the screw axis, the magnitude of this vector is the angle over which the body is rotated, and the second three-vector represents the translation of the body along the screw axis. However, in accordance with the *structural* fact that finite displacements belong to SE(3) and not to the vector space se(3) (Sects 3.3, 3.4), this representation lacks the addition property of screws, velocity twists or infinitesimal displacements twists: the addition of $\mathbf{t}_d^1$ and $\mathbf{t}_d^2$ is *not* the finite displacement twist that represents the composition of the finite displacements represented by $\mathbf{t}_d^1$ and $\mathbf{t}_d^2$. Hence, $\mathbf{t}_d$ is *not* a screw; we only called it "screw-like" because it has similarly looking coordinates.

Wrench. *Wrench coordinates* are also given by six-vectors $(\boldsymbol{f}^T \, \boldsymbol{m}^T)^T$. The interpretation is as follows: the three-vector $\boldsymbol{f}$ represents the coordinates of the linear force, expressed with respect to the world reference frame; the three-vector $\boldsymbol{m}$ is the sum of (i) the pure torques working on the body, and (ii) the moment of $\boldsymbol{f}$ with respect to the origin of the world reference frame. "Screw" and "pose" interpretations apply here too: $\boldsymbol{m}$ depends on the chosen reference point. The pitch of a wrench is the ratio $\boldsymbol{m}/\boldsymbol{f}$; it has the units of length.

4.5.3 Importance of screws

Typical undergraduate textbooks rely on *vectors* to describe the physics of a moving point mass. The most important vector properties are: (i) *addition* is defined and meaningful (e.g., the sum of two translational velocity vectors is a vector that represents the combined velocity), (ii) the *scalar product* (or "dot product" "·") of two vectors is a well-defined scalar (e.g., force times displacement is energy), and (iii) the *vector product* (or "cross product" "×") of two vectors is a well-defined vector. For example, the vector product of an angular velocity with a moment arm vector is the translational velocity of the end point of the moment arm vector. The concepts and properties of *screws* are usually not treated. However, screws have the three above-mentioned vector properties too, Chapter 3, with the *spatial scalar product*, or "pairing," replacing the three-vector scalar product, and the *motor product*, [5, 26, 27], (also called the *spatial cross product*, [10], or Lie bracket, Sect. 3.4) replacing the three-vector cross product. Brand [5] proved that, in terms of this three-vector cross product, the motor product is given by

$$\mathbf{t}^1 \times \mathbf{t}^2 \triangleq \begin{pmatrix} \boldsymbol{\omega}^1 \times \boldsymbol{\omega}^2 \\ \boldsymbol{v}^1 \times \boldsymbol{\omega}^2 - \boldsymbol{v}^2 \times \boldsymbol{\omega}^1 \end{pmatrix}. \tag{4.16}$$

Note that (i) we use the same symbol "$\times$" for both the cross and motor products, an (ii) the motor product in Eq. (4.16) has indeed all the properties of the Lie algebra described in Sect. 3.4, such as anti-symmetry and the Jacobi identity. Roughly speaking,

> **Fact-to-Remember 20 (Screws and vectors)**
> *Screws are for rigid bodies what vectors are for point masses.*

Duality. The following Chapters extensively use twists and wrenches, as the basic motion and force concepts in the kinematics and dynamics of rigid bodies.

> **Fact-to-Remember 21 (Duality twist–wrench)**
> *Twist and wrenches are both "built on top of" the same geometrical concept of the screw. Hence, whenever we can derive a result about twists and this result depends only on the geometric properties of the underlying screws, then we have immediately derived a* <u>*dual*</u> *result for the wrenches built with the same screws.*

Such dualities occur very often in the kinematics of serial and parallel robot arms. The next Chapters contain many examples of these dualities.

4.5.4 Reciprocity of screws

Reciprocity as defined in Sect. 3.9 is not in the first place a property of twists and wrenches, but a property of *screws*. Two screws $\mathbf{s}_1 = (\boldsymbol{d}_1^T\ \boldsymbol{c}_1^T)^T$ and $\mathbf{s}_2 = (\boldsymbol{d}_2^T\ \boldsymbol{c}_2^T)^T$ are reciprocal if

$$\mathbf{s}_1 \widetilde{\boldsymbol{\Delta}} \mathbf{s}_2 = \boldsymbol{d}_1^T \boldsymbol{c}_2 + \boldsymbol{d}_2^T \boldsymbol{c}_1 = 0, \tag{4.17}$$

with

$$\boxed{\widetilde{\boldsymbol{\Delta}} \triangleq \begin{pmatrix} \boldsymbol{0}_3 & \boldsymbol{1}_3 \\ \boldsymbol{1}_3 & \boldsymbol{0}_3 \end{pmatrix}.} \tag{4.18}$$

In robotics, one often needs to calculate the reciprocal set of a set of screws. For example, a five degrees of freedom robot has a one-dimensional reciprocal set, consisting of all wrenches exerted on the robot's end effector that can be taken up completely by the mechanical structure of the robot, i.e., without requiring any power from the motors. From a numerical point of view, the calculation of the reciprocal set of the set of screws $\{\mathbf{s}^1, \dots, \mathbf{s}^n\}$ looks a lot like the calculation of its orthogonal complement. Indeed, the "classical" Gram-Schmidt procedure, [12], is applicable, if one first premultiplies all screws in $\{\mathbf{s}^1, \dots, \mathbf{s}^n\}$ by the matrix $\widetilde{\boldsymbol{\Delta}}$, Eq. (4.18).

References for this Chapter

[1] J. Angeles. *Rational Kinematics*. Springer Verlag, 1988.

[2] R. S. Ball. *Theory of screws: a study in the dynamics of a rigid body*. Hodges, Foster and Co, Dublin, Ireland, 1876.

[3] R. Bernhardt and S. L. Albright. *Robot Calibration*. Chapman & Hall, 1993.

[4] A. Bottema and B. Roth. *Theoretical Kinematics*. Dover Books on Engineering. Dover Publications, Inc., Mineola, NY, 1990.

[5] L. Brand. *Vector and tensor analysis*. Wiley, New York, NY, 1948. 3rd reprint.

[6] W. L. Burke. *Applied differential geometry*. Cambridge University Press, 1992.

[7] A. Cayley. On a new analytical representation of curves in space. *Quarterly Journal of Pure and Applied Mathematics*, 3:225–236, 1860.

[8] W. K. Clifford. Preliminary sketch of biquaternions. *Proceedings of the London Mathematical Society*, 4(64, 65):381–395, 1873. Reprinted in *Mathematical Papers*, Chelsea Publishing Company, New York, pp. 189–200, 1968.

[9] J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Trans. ASME J. Appl. Mech.*, 23:215–221, 1955.

[10] R. Featherstone. *Robot dynamics algorithms*. Kluwer Academic Publishers, Boston, MA, 1987.

[11] H. Goldstein. *Classical mechanics*. Addison-Wesley Series in Physics. Addison-Wesley, Reading, MA, 2nd edition, 1980.

[12] G. Golub and C. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1989.

[13] W. R. Hamilton. *Lectures on Quaternions*. Hodges and Smith, Dublin, Ireland, 1853.

[14] R. S. Hartenberg and J. Denavit. *Kinematic synthesis of linkages*. McGraw-Hill, New York, NY, 1964.

[15] S. A. Hayati and M. Mirmirani. Improving the absolute positioning accuracy of robot manipulators. *J. Robotic Systems*, 2(4):397–441, 1985.

[16] K. H. Hunt. *Kinematic Geometry of Mechanisms*. Oxford Science Publications, Oxford, England, 2nd edition, 1990.

[17] A. Karger and J. Novak. *Space kinematics and Lie groups*. Gordon and Breach, New York, NY, 1985.

[18] J. Lončarić. *Geometrical Analysis of Compliant Mechanisms in Robotics*. PhD thesis, Harvard University, Cambridge, MA, 1985.

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

[20] F. C. Park. Distance metrics on the rigid-body motions with applications to mechanism design. *Trans. ASME J. Mech. Design*, 117:48–54, 1995.

[21] J. Plücker. On a new geometry of space. *Philosophical Transactions of the Royal Society of London*, 155:725–791, 1865.

[22] J. Plücker. Fundamental views regarding mechanics. *Philosophical Transactions of the Royal Society of London*, 156:361–380, 1866.

[23] K. S. Roberts. A new representation for a line. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 635–640, Ann Arbor, MI, 1988.

[24] K. Schröer, S. L. Albright, and M. Grethlein. Complete, minimal and model-continuous kinematic models for robot calibration. *Rob. Comp.-Integr. Manufact.*, 13(1):73–85, 1997.

[25] L. Van Aken. *Robot motions in free space: task specification and trajectory planning*. PhD thesis, Katholieke Universiteit Leuven, Dept. Werktuigkunde, 1987.

[26] R. von Mises. Motorrechnung, ein neues Hilfsmittel der Mechanik. *Zeitschrift für Angewandte Mathematik und Mechanik*, 4(2):155–181, 1924. English translation by E.J. Baker and K. Wohlhart, published by the Institute for Mechanics, University of Technology, Graz, Austria, 1996.

[27] K. Wohlhart. Motor tensor calculus. In J.-P. Merlet and B. Ravani, editors, *Computational Kinematics '95*, pages 93–102, Dordrecht, 1995. Kluwer Academic Publishers.

[28] A. T. Yang. Calculus of screws. In W. R. Spillers, editor, *Basic questions of design theory*, pages 265–281. North Holland, Amsterdam, 1974.

[29] M. S. C. Yuan and F. Freudenstein. Kinematic analysis of spatial mechanisms by means of screw coordinates. Part 1—Screw coordinates. *Trans. ASME J. Eng. Industry*, 93:61–66, 1971.

[30] H. Zhuang, Z. S. Roth, and F. Hamano. A complete and parametrically continuous kinematic model for robot manipulators. *IEEE Trans. Rob. Automation*, 8(4):451–463, 1992.

Chapter 5

Orientation coordinates

5.1 Introduction

Imagine a set of *orthogonal* and *right-handed* reference frames, all having the same point as origin. These frames represent different *orientations* of the rigid body to which they are fixed. The orientation of a frame is not an absolute concept, since it implies a second reference frame with respect to which it is defined. Hence, one should speak only about the *relative orientation* between two frames. Orientation and its time derivatives, i.e., angular velocity and acceleration, are quite distinct from the more intuitive concepts of Euclidean position and its time derivatives. The properties of the corresponding coordinate representations will reflect this *structural* difference.

Fact-to-Remember 22 (Basic ideas of this Chapter)
"Rotation" ("(relative) orientation") is the main difference between the kinematics of points and the kinematics of rigid bodies. A rigid body has three degrees of freedom in orientation. However, every representation with only three parameters inevitably has coordinate singularities at a number of orientations. The second important fact is that rotational velocity is not found as the time derivative of any representation of orientation; however, integrating factors always exist. Hence, orientation coordinates are integrable (or "holonomic").

A coordinate singularity occurs whenever a small change in the represented system (i.c., orientation) cannot be represented by a small change in coordinates. This concept will appear many more times in this text.

5.2 Rotation matrix

Orientation and rotation are related concepts. They represent the relative pose of two (reference rames on) rigid bodies, *modulo* the translation: choose an arbitrary reference point on both bodies, and translate all points in the second body over the (inverse of the) vector connecting both points; what remains of the relative displacement is the relative orientation of both bodies. This Section describes rotation matrices as appropriate and very often used mathematical representations of relative orientation; later Sections introduce alternative representations.

5.2.1 Definition and use

Several coordinate representations exist to express the relative orientation of a frame $\{b\}$ with respect to a frame $\{a\}$. The 3×3 *rotation matrix* ${}^{b}_{a}\boldsymbol{R}$ is among the most popular. Other names for this matrix are *orientation matrix*, or *matrix of direction cosines*.

Fact-to-Remember 23 (Rotation matrix)
The columns of ${}^{b}_{a}\boldsymbol{R}$ *contain the components of the unit vectors* $\boldsymbol{x}^b, \boldsymbol{y}^b$ *and* $\boldsymbol{z}^b$ *along the axes of frame* $\{b\}$*, expressed in the reference frame* $\{a\}$ *(Fig. 5.1):*

$$
{}^{b}_{a}\boldsymbol{R} = \left(R_{ij}\right) = \begin{pmatrix} {}_{a}\boldsymbol{x}^b & {}_{a}\boldsymbol{y}^b & {}_{a}\boldsymbol{z}^b \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}^b \cdot \boldsymbol{x}^a & \boldsymbol{y}^b \cdot \boldsymbol{x}^a & \boldsymbol{z}^b \cdot \boldsymbol{x}^a \\ \boldsymbol{x}^b \cdot \boldsymbol{y}^a & \boldsymbol{y}^b \cdot \boldsymbol{y}^a & \boldsymbol{z}^b \cdot \boldsymbol{y}^a \\ \boldsymbol{x}^b \cdot \boldsymbol{z}^a & \boldsymbol{y}^b \cdot \boldsymbol{z}^a & \boldsymbol{z}^b \cdot \boldsymbol{z}^a \end{pmatrix}. \tag{5.1}
$$

${}_{a}\boldsymbol{x}^b$ *is the notation for the three-vector with the coordinates of the end-point of the unit vector* $\boldsymbol{x}^b$ *in the reference frame* $\{a\}$*, formed by the three unit vectors* $\boldsymbol{x}^a, \boldsymbol{y}^a$ *and* $\boldsymbol{z}^a$*.*

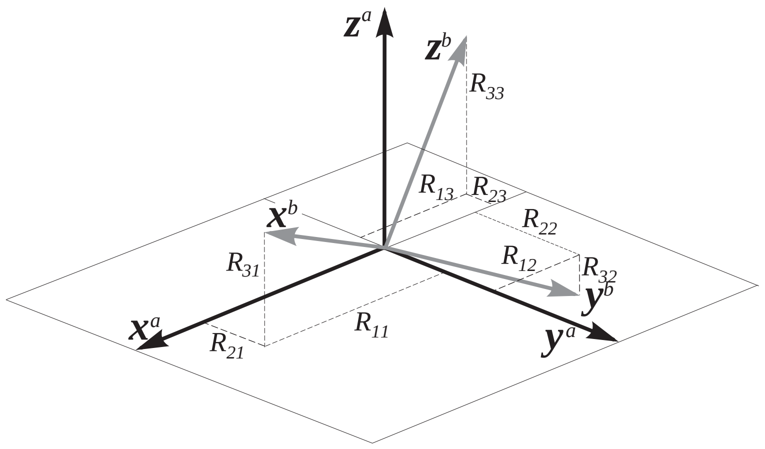


Figure 5.1: Components R_{ij} of a rotation matrix ${}^{b}_{a}\boldsymbol{R}$.

Equation 5.1 allows to calculate the coordinates of a point $\boldsymbol{p}$ with respect to the frame $\{a\}$ if the coordinates of this *same* point $\boldsymbol{p}$ are known with respect to the frame $\{b\}$ (*and* if $\{a\}$ and $\{b\}$ have the same origin!):

$$
{}_{a}\boldsymbol{p}_x = \boldsymbol{p} \cdot \boldsymbol{x}^a = \left({}_{b}\boldsymbol{p}_x\, \boldsymbol{x}^b + {}_{b}\boldsymbol{p}_y\, \boldsymbol{y}^b + {}_{b}\boldsymbol{p}_z\, \boldsymbol{z}^b\right) \cdot \boldsymbol{x}^a.
$$

Hence

$$
{}_{a}\boldsymbol{p} = {}^{b}_{a}\boldsymbol{R}\; {}_{b}\boldsymbol{p}, \quad \text{or} \quad \begin{pmatrix} {}_{a}\boldsymbol{p}_x \\ {}_{a}\boldsymbol{p}_y \\ {}_{a}\boldsymbol{p}_z \end{pmatrix} = {}^{b}_{a}\boldsymbol{R} \begin{pmatrix} {}_{b}\boldsymbol{p}_x \\ {}_{b}\boldsymbol{p}_y \\ {}_{b}\boldsymbol{p}_z \end{pmatrix}. \tag{5.2}
$$

Note the notational convention: subscript "b" on coordinate vector ${}_{b}\boldsymbol{p}$ "cancels" with superscript "b" on ${}^{b}_{a}\boldsymbol{R}$, and is replaced by subscript "a."

noa* notation** In robotics, one sometimes encounters the notation ***noa for the three axes of a right-handed orthogonal reference frame. This nomenclature is due to Richard Paul, [20], which introduced it to describe the different axes of an end-effector frame attached to a parallel-jaw gripper of the robot. (A "parallel-jaw" gripper is probably the oldest and most frequent type of "robotic hand"; it simply consists of two parallel plates that can open and close.) In this context, these three letters stand for:

"**o**pen" direction: the direction in which the fingers of the gripper open and close. This direction is normal to the gripper plates.

"**a**pproach" direction: the direction in which the robot gripper approaches its target. This direction is parallel to the "finger direction" of the gripper plates.

"**n**ormal" direction. The direction orthogonal to the previous two directions.

5.2.2 Rotations about frame axes

Rotations about the frame axes have simple expressions. Let $\boldsymbol{R}(X, \alpha)$ denote the rotation mapping that moves the endpoint of the vector $\boldsymbol{p}$ over a circular arc of α radians to a vector $\boldsymbol{p}'$ (see Fig. 5.2, where $\boldsymbol{p} = \boldsymbol{e}_y$ or $\boldsymbol{e}_z$), and during which the centre of the arc lies on the X axis. Hence, the arc itself lies in a plane through $\boldsymbol{p}$ and orthogonal to X. The angle is oriented according to the right-hand rule about the X axis.

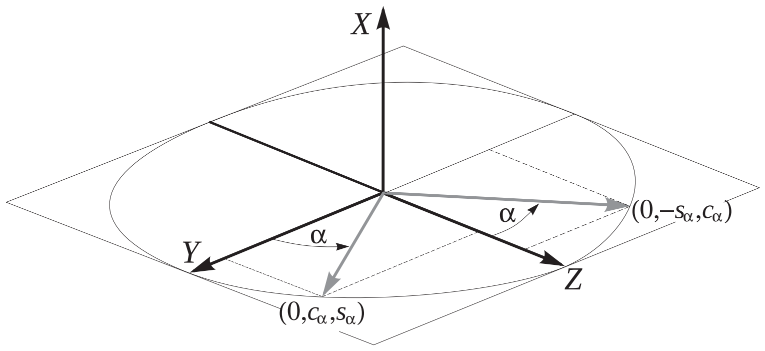


Figure 5.2: Rotation over an angle α about the X axis. c_α stands for $\cos(\alpha)$, and s_α stands for $\sin(\alpha)$.

The rotation matrix of this rotation is easily derived from Fig. 5.2:

$$\boldsymbol{R}(X, \alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix}. \tag{5.3}$$

This means that the components $(\boldsymbol{p}_x \; \boldsymbol{p}_y \; \boldsymbol{p}_z)^T$ of any vector $\boldsymbol{p}$ are mapped to the components $(\boldsymbol{p}'_x \; \boldsymbol{p}'_y \; \boldsymbol{p}'_z)^T$ as

$$\begin{pmatrix} \boldsymbol{p}'_x \\ \boldsymbol{p}'_y \\ \boldsymbol{p}'_z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_x \\ \boldsymbol{p}_y \\ \boldsymbol{p}_z \end{pmatrix}. \tag{5.4}$$

The rotation matrices $\boldsymbol{R}(Y,\alpha)$ and $\boldsymbol{R}(Z,\alpha)$, corresponding to rotations about the Y and Z frame axes respectively, are found in a similar way:

$$\boldsymbol{R}(Y,\alpha) = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix}, \qquad \boldsymbol{R}(Z,\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{5.5}$$

5.2.3 Active and passive interpretation

Besides a (passive) transformation of a vector's coordinates in a frame $\{b\}$ to its coordinates in a frame $\{a\}$, a rotation mapping R has also a second, equally important interpretation: R *moves* the frame $\{a\}$ into the frame $\{b\}$. We call this interpretation the *active* interpretation of the rotation mapping. It moves the unit vector $\boldsymbol{x}^a$ that lies along the X axis of $\{a\}$ to the unit vector $\boldsymbol{x}^b$ that lies along the X axis of $\{b\}$: $R(\boldsymbol{x}^a) = \boldsymbol{x}^b$. Similarly for the unit vectors along Y and Z. Hence, in matrix form:

$$\begin{pmatrix} {}_a\boldsymbol{x}^b & {}_a\boldsymbol{y}^b & {}_a\boldsymbol{z}^b \end{pmatrix} = \boldsymbol{R} \begin{pmatrix} {}_a\boldsymbol{x}^a & {}_a\boldsymbol{y}^a & {}_a\boldsymbol{z}^a \end{pmatrix} = \boldsymbol{R}. \tag{5.6}$$

Equation (5.1) implies that $\boldsymbol{R} = {}^b_a\boldsymbol{R}$, i.e., active and passive interpretations of orientation and rotation have the same matrix representation.

Fact-to-Remember 24 (Active and passive interpretations)
A rotation mapping has both an <u>active</u> and a <u>passive</u> interpretation, [17]: the passive form transforms coordinates of the <u>same</u> spatial point (or vector) from one reference frame to another (i.e., it represents "orientation") while the active interpretation moves one spatial point (or vector) to <u>another</u> spatial point (or vector) (i.e., it represents "rotation"). Both interpretations are represented by the same matrix.

5.2.4 Uniqueness

From the above-mentioned construction of the rotation matrix, the following fact is obvious, but important:

Fact-to-Remember 25 (Uniqueness)
A rotation matrix is a <u>unique</u> and <u>unambiguous</u> representation of the relative orientation of two right-handed, orthogonal reference frames in the Euclidean space E^3. This means that one single rotation matrix corresponds to each relative orientation, and each rotation matrix represents one single relative orientation.

Note that many mechanics or geometry books (e.g., [5, 7, 9, 14]) use another definition for the rotation matrix: this alternative corresponds to the *transpose* of the rotation matrices used in this text. Moreover, some older references use *left-handed* reference frames. Be aware of these alternative definitions when you consult such references! Fortunately, all modern robotics literature adheres to the same convention as this text.

5.2.5 Inverse

The *inverse* of a rotation matrix ${}^b_a\boldsymbol{R}$ is, by definition, ${}^a_b\boldsymbol{R}$ since it transforms coordinates with respect to $\{a\}$ into coordinates with respect to $\{b\}$. The defining equation (5.1) gives

$$
{}^a_b\boldsymbol{R} = \begin{pmatrix} \boldsymbol{x}^a \cdot \boldsymbol{x}^b & \boldsymbol{y}^a \cdot \boldsymbol{x}^b & \boldsymbol{z}^a \cdot \boldsymbol{x}^b \\ \boldsymbol{x}^a \cdot \boldsymbol{y}^b & \boldsymbol{y}^a \cdot \boldsymbol{y}^b & \boldsymbol{z}^a \cdot \boldsymbol{y}^b \\ \boldsymbol{x}^a \cdot \boldsymbol{z}^b & \boldsymbol{y}^a \cdot \boldsymbol{z}^b & \boldsymbol{z}^a \cdot \boldsymbol{z}^b \end{pmatrix}. \tag{5.7}
$$

Comparing Eqs (5.1) and (5.7) yields

$$
\boxed{{}^a_b\boldsymbol{R} = {}^b_a\boldsymbol{R}^T,} \tag{5.8}
$$

and

Fact-to-Remember 26 (Orthogonality)
Each rotation matrix is an orthogonal linear transformation (i.e., it is an orthogonal matrix), since it satisfies the following six orthogonality constraints:

$$
\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{1}_3. \tag{5.9}
$$

This implies that

$$
\boxed{\boldsymbol{R}^{-1} = \boldsymbol{R}^T.} \tag{5.10}
$$

5.2.6 Non-minimal representation

A direct consequence of Eq. (5.9) is that

Fact-to-Remember 27 (Non-minimal representation)
A rotation matrix is a non-minimal representation of an orientation, since it uses nine numbers to represent three degrees of freedom. The orthogonality constraints (5.9) uniquely determine the six dependent parameters. One of its big advantages is that it has no coordinate singularities; this will not be the case for any of the minimal representations discussed later.

5.2.7 Isometry—SO(3)

Rotations are *isometries* of the Euclidean space, since they maintain angles between vectors, and lengths of vectors. Moreover, right-handed frames are mapped into right-handed frames. So, the determinant of rotation matrices is +1. That's why mathematicians call them "orientation preserving (or special) orthogonal linear transformations," or "proper orthogonal matrices." Their *structural* properties are described by the Lie group SO(3), Sect. 3.6.

5.2.8 Composition of rotations

Rotation matrices are faithful representations of SO(3). Hence, *composition* of rotations is represented by *multiplication* of rotation matrices. The following paragraphs derive the exact formula from reasoning with the active interpretation of rotation matrices.

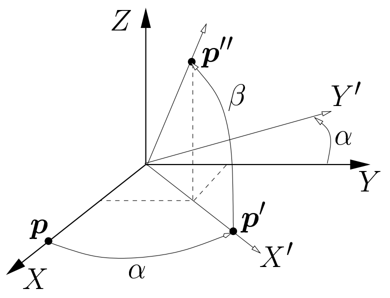


Figure 5.3: Rotation over an angle α about the Z axis, followed by a rotation about the moved Y-axis (i.e., Y') over an angle $-\beta$.

Figure 5.3 shows the rotation of the unit vector $\boldsymbol{p}$ on the X-axis to the point $\boldsymbol{p}'$, due to a rotation about Z over an angle α. Then, $\boldsymbol{p}'$ is moved to $\boldsymbol{p}''$, by a rotation over an angle β about the Y'-axis, i.e., the axis to which the Y-axis is moved by the rotation over α about Z. (In fact, the rotation about Y' is over the angle $-\beta$, due to the right-hand rule convention.) It is easy to calculate that $\boldsymbol{p}''$ has the following coordinates in the original frame $\{XYZ\}$:

$$\boldsymbol{p}'' = \begin{pmatrix} c_\alpha c_\beta \\ s_\alpha c_\beta \\ s_\beta \end{pmatrix}, \tag{5.11}$$

where $c_\alpha = \cos(\alpha)$, etc. The coordinates of the rotated unit vectors along the Y and Z axes can be calculated in a similar way. Bringing these three results together gives the rotation matrix $\boldsymbol{R}(ZY, \alpha, -\beta)$ corresponding to the composition of, first, $\boldsymbol{R}(Z, \alpha)$, the rotation about Z over the angle α, and, then, $\boldsymbol{R}(Y, -\beta)$, the rotation about Y' (i.e., the *moved* Y-axis) over the angle $-\beta$:

$$\boxed{\boldsymbol{R}(ZY, \alpha, -\beta) = \begin{pmatrix} c_\alpha c_\beta & -s_\alpha & -c_\alpha s_\beta \\ s_\alpha c_\beta & c_\alpha & -s_\alpha s_\beta \\ s_\beta & 0 & c_\beta \end{pmatrix} = \boldsymbol{R}(Z, \alpha)\boldsymbol{R}(Y, -\beta).} \tag{5.12}$$

Somewhere around 1840, [4, 23], the French mathematician Olinde Rodrigues (1794–1851) seems to have been the first to find the coordinate expressions for composing rotations this way.

Inverse. The *inverse* of a *single* rotation about an axis equals the rotation about the same axis, but over the negative of the rotation angle:

$$\boldsymbol{R}^{-1}(X, \alpha) = \boldsymbol{R}(X - \alpha). \tag{5.13}$$

The inverse of a compound orientation follows immediately from the rule for the inverse of the matrix product:

$$\boldsymbol{R}^{-1}(ZY, \alpha, -\beta) = \boldsymbol{R}(Y, \beta)\boldsymbol{R}(Z, -\alpha). \tag{5.14}$$

5.2.9 Rotation matrix time rate—Angular velocity

Robot devices are often *position controlled*, i.e., the user commands the robot to move to a given position, and the robot control software should do its best to attain this position as reliably and accurately as possible. However, many applications require *velocity control*; e.g., spray painting or applying a continuous stream of glue to an automobile window seam. Hence, a representation of velocity is needed too. Similarly to the case of points, the velocity of a rigid body is calculated as the differential motion between two nearby poses in a given small period of time. For the translational velocity $\boldsymbol{v}$ of (a reference point on) the moving body, this calculation is performed straightforwardly by the classical difference relation between two nearby *position vectors* $\boldsymbol{p}(t^1)$ and $\boldsymbol{p}(t^2)$:

$$\boldsymbol{v} = \frac{d\boldsymbol{p}}{dt} \approx \frac{\boldsymbol{p}(t^2) - \boldsymbol{p}(t^1)}{t^2 - t^1}. \tag{5.15}$$

However, the relationship between the time rate of the orientation matrix on the one hand, and the angular velocity three-vector $\boldsymbol{\omega}$ on the other hand, is a bit more complicated:

1. The coordinates with respect to $\{a\}$ of a point fixed to $\{b\}$ are:

$$_a\boldsymbol{p}(t) = {}^b_a\boldsymbol{R}(t)\, {}_b\boldsymbol{p}. \tag{5.16}$$

2. Assume that reference frame $\{b\}$ in Eq. (5.2) moves with respect to $\{a\}$ with angular velocity $\boldsymbol{\omega}$. Hence, the rotation matrix ${}^b_a\boldsymbol{R}$ changes. Since the point $\boldsymbol{p}$ is rigidly fixed to the frame $\{b\}$, its components ${}_b\boldsymbol{p}$ with respect to $\{b\}$ do not change.

3. The time derivative of Eq. (5.16) gives the instantaneous translational velocity of the endpoint of the position vector $\boldsymbol{p}$:

$$\begin{aligned} {}_a\dot{\boldsymbol{p}} &= {}^b_a\dot{\boldsymbol{R}}\ {}_b\boldsymbol{p} \\ &= {}^b_a\dot{\boldsymbol{R}}\ \left({}^a_b\boldsymbol{R}\ {}_a\boldsymbol{p}\right). \end{aligned} \tag{5.17}$$

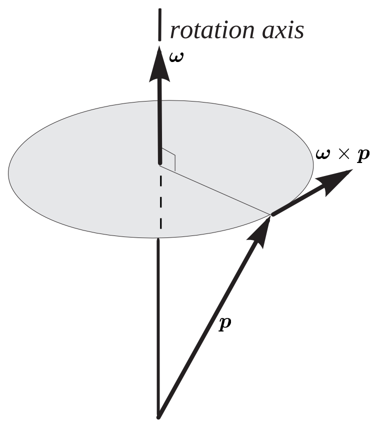


Figure 5.4: Translational velocity of a point fixed to a rigid body rotating with an angular velocity $\boldsymbol{\omega}$.

4. Alternatively, Fig. 5.4 shows that the same translational velocity is given by:

$$\begin{aligned}\dot{\boldsymbol{p}} &= \boldsymbol{\omega} \times \boldsymbol{p}\\ &= [\boldsymbol{\omega}]\, \boldsymbol{p}.\end{aligned} \tag{5.18}$$

$[\boldsymbol{\omega}]$ is the skew-symmetric matrix operator that represents the vector product "$[\boldsymbol{\omega}]\,\cdot$" with the three-vector $\boldsymbol{\omega}$:

$$[\boldsymbol{\omega}] \triangleq \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}. \tag{5.19}$$

Hence, the following linear relationships result from Eqs (5.17) and (5.18):

$$\boxed{[{}_a\boldsymbol{\omega}] = {}^b_a\dot{\boldsymbol{R}}\ {}^b_a\boldsymbol{R}^{-1}, \quad \text{or} \quad {}^b_a\dot{\boldsymbol{R}} = [{}_a\boldsymbol{\omega}]\ {}^b_a\boldsymbol{R}.} \tag{5.20}$$

The angular velocity vector and the time rate of the rotation matrix are coupled by the inverse of the current rotation matrix, which acts as a so-called *integrating factor*, [22]. This existence of an integrating factor is a *property* of rotations, i.e., *all* orientation representations will have this property. A related property is *holonomy*: executing a "closed trajectory" (i.e., one stops where one has started) in "rotation matrix space" leads to a closed trajectory in orientation space. Later Chapters will present some *non-holonomic* robotic systems.

5.2.10 Exponential and logarithm

Section 3.5 introduced the concept of *exponentiating* a velocity to get a change in pose. This Section gives a coordinate representation for this mapping in the case of an *angular velocity* about a frame axis, [25]. Assume a constant angular velocity $\boldsymbol{\omega} = (\omega_x\ 0\ 0)^T$ along the X axis of the base reference frame. After time t, the frame is rotated over an angle $\alpha_t = \omega_x t$ radians, and the corresponding rotation matrix is, Eq. (5.3),

$$\boldsymbol{R}(X, \alpha_t) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha_t) & -\sin(\alpha_t) \\ 0 & \sin(\alpha_t) & \cos(\alpha_t) \end{pmatrix}.$$

The name "exponential" for this operation becomes clear by noticing that the matrix $\boldsymbol{R}(X, \alpha_t)$ is equal to the *matrix exponential* of the skew-symmetric matrix $[\boldsymbol{\omega}]$ that corresponds to the angular velocity $\boldsymbol{\omega}$, Eq. (5.19): the solution of Eq. (5.20) is $\boldsymbol{R}(t) = C \exp\left([\boldsymbol{\omega}]t\right), C \in \mathbb{R}$. The angular motion starts with $\boldsymbol{R}(t=0)$ at time $t = 0$, such that $\exp\left([\boldsymbol{\omega}]t\right) = I_{3\times 3}$ and thus $C = 1$. Hence,

$$\boxed{\boldsymbol{R}(X, \alpha_t) = \exp\left([\boldsymbol{\omega}]t\right).} \tag{5.21}$$

An alternative, coordinate-based proof consists of filling in

$$\boldsymbol{A} \triangleq [\boldsymbol{\omega}]t = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -\omega_x t \\ 0 & \omega_x t & 0 \end{pmatrix},$$

in the Taylor series of the matrix exponential:

$$\exp(\boldsymbol{A}) \triangleq \mathbf{1} + \boldsymbol{A} + \frac{\boldsymbol{A}^2}{2!} + \frac{\boldsymbol{A}^3}{3!} + \ldots \tag{5.22}$$

The matrix powers of $\boldsymbol{A}$ are

$$\boldsymbol{A}^2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\omega_x^2 t^2 & 0 \\ 0 & 0 & -\omega_x^2 t^2 \end{pmatrix}, \quad \boldsymbol{A}^3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -\omega_x^3 t^3 \\ 0 & \omega_x^3 t^3 & 0 \end{pmatrix} = -\omega_x^2 t^2 \boldsymbol{A}. \tag{5.23}$$

Hence, all higher powers of $\boldsymbol{A}$ are proportional to $\boldsymbol{A}$ or $\boldsymbol{A}^2$, and the proportionality factors correspond exactly to the Taylor series of the sines and cosines of α_t used in $\boldsymbol{R}(X, \alpha_t)$. The reasoning above holds for *general* rotations too, and not just for rotations about frame axes.

Logarithm The previous paragraphs proved that the exponential of an angular velocity about a frame axis yields a finite rotation. The converse is also true: each finite rotation about, for example, the X axis can be generated by applying an angular velocity about the X axis during one unit of time. This velocity is called the *logarithm* of the finite rotation.

5.2.11 Infinitesimal rotation

Equations (5.21) and (5.22) give an easy way to find the first order approximation of a small rotation about a given axis: let $(\delta_x \ \delta_y \ \delta_z)^T$ be a small rotation about the frame axes (or, equivalently, an angular velocity applied during a small time period), then stopping the Taylor series in Eq. (5.22) after the linear term gives:

$$\boxed{\boldsymbol{R}_\Delta = \begin{pmatrix} 1 & -\delta_z & \delta_y \\ \delta_z & 1 & \delta_x \\ -\delta_y & \delta_x & 1 \end{pmatrix}.} \tag{5.24}$$

$\boldsymbol{R}_\Delta$ is called an *inifinitesimal rotation matrix.* (This text will use the subscript "Δ" many more times to denote infinitesimal quantities.)

5.3 Euler angles

The previous Section introduced the rotation matrix as mathematical representation for relative orientation; this Section looks at minimal representations, i.e., sets of only three numbers. These numbers are called *Euler angles.* They describe any orientation as a sequence of three rotations about *moving* frame axes, i.e., the second rotation takes place about an axis in the frame *after* it was moved by the first rotation, and so on. Euler angles are extensively used in robotics, but also in many other disciplines.

5.3.1 Euler's contributions

Motion of rigid bodies, and especially rotational motion, was a primary source of inspiration for all mathematicians of the eighteenth and nineteenth centuries, even though they were "just" looking for a intuitive application for their work on mathematical analysis or geometry. The name of the Swiss mathematician Leonhard Euler (1707–1783) is intimately connected to the following theorems that are fundamental for the kinematics of robotic devices and robotic tasks (but he surely was neither the first one, nor the only one to work on these problems!). Euler's results are the theoretical basis for

Fact-to-Remember 28 (Minimal representations)
Around 1750, Euler proved that the relative orientation of two coordinate systems can be specified by a set of three independent angles, [7, 21].

Fact-to-Remember 29 (Euler's Theorem, 1775)
Any displacement of a rigid body that leaves one point fixed is a rotation about some axis, [1, 2, 7].

The first theorem results in the large set of three-angle orientation representations (all of them are called *Euler angles*!) discussed in this Section.

The second theorem predicts the existence of a so-called *equivalent axis* and the corresponding *equivalent rotation angle*, discussed in more detail in Section 5.4. "Displacement" means that only the initial and final poses of the body are taken into account, *not* the trajectory the body followed to move between those poses.

5.3.2 Composition of rotations about moving axes

One of the major characteristics of robotic devices is that they consist of multiple bodies connected by joints. Each joint between two links contributes to the orientation of the robot's "end effector" (unless the joint is prismatic!). The total end effector orientation is the *composition* of these individual contributions. Section 5.2.8 explained already how to compose two subsequent rotations; this section uses these results to find the rotation matrices generated by different sets of three Euler angles. These are illustrated by means of the example of a simple serial kinematic chain with three revolute joints (Fig. 5.5). This kinematic structure is commonly used for "wrists" of serial manipulators, i.e., that part of the robot arm that takes care of the final orientation of the end effector. It consists of three revolute joints, whose axes intersect in one single point. The structure has an immobile base (the "zeroth" link), to which a base reference frame $\{bs\}$ is attached. The first joint rotates about the Z axis of $\{bs\}$, and moves the first, second and third link of the wrist together with respect to $\{bs\}$. A reference frame $\{a\}$ is attached to the first link. Similarly, the second joint rotates about the X axis of $\{a\}$, and moves a frame $\{b\}$ on the second link with respect to $\{a\}$. Finally, the third joint rotates about the Z axis of $\{b\}$, and moves the end effector frame $\{ee\}$ with respect to $\{b\}$. The above-mentioned conventions explain why this kinematic structure is referred to as a *ZXZ* wrist.

Forward mapping. Obviously, it must be possible to obtain the relative orientation of the "end effector" frame $\{ee\}$ with respect to the "base" frame $\{bs\}$ from the relative orientation of $\{ee\}$ with respect to $\{b\}$, $\{b\}$ with respect to $\{a\}$, and $\{a\}$ with respect to $\{bs\}$. With respect to their local reference frames, each of these one-joint transformations has the simple form of the frame axis rotation matrices in Eqs (5.3) and (5.5), i.e., ${}^{ee}_{b}\boldsymbol{R} = \boldsymbol{R}(Z,\gamma), {}^{b}_{a}\boldsymbol{R} = \boldsymbol{R}(X,\beta), {}^{a}_{bs}\boldsymbol{R} = \boldsymbol{R}(Z,\alpha)$, respectively. The angles α, β and γ are the joint angles of the three revolute joints, with respect to the "zero" configuration in which $\{ee\}$ and $\{bs\}$ are parallel (Fig. 5.5). The total resulting rotation matrix ${}^{ee}_{bs}\boldsymbol{R}$ is found from applying Eq. (5.12) twice. That equation was derived with the *active* interpretation of rotations; here, an alternative derivation is constructed using the *passive* interpretation:

1. The unit vector $\boldsymbol{x}^{ee}$ along the X axis of the end effector reference frame has coordinates ${}_{b}\boldsymbol{x}^{ee} = {}^{ee}_{b}\boldsymbol{R}\ {}_{ee}\boldsymbol{x}^{ee} = {}^{ee}_{b}\boldsymbol{R}\ (1\,0\,0)^{T}$ in the reference frame $\{b\}$. This is a straightforward application of the definition Eq. (5.1).
2. These coordinates ${}_{b}\boldsymbol{x}^{ee}$, in turn, are transformed into ${}_{a}\boldsymbol{x}^{ee} = {}^{b}_{a}\boldsymbol{R}\ {}_{b}\boldsymbol{x}^{ee}$, with respect to frame $\{a\}$.

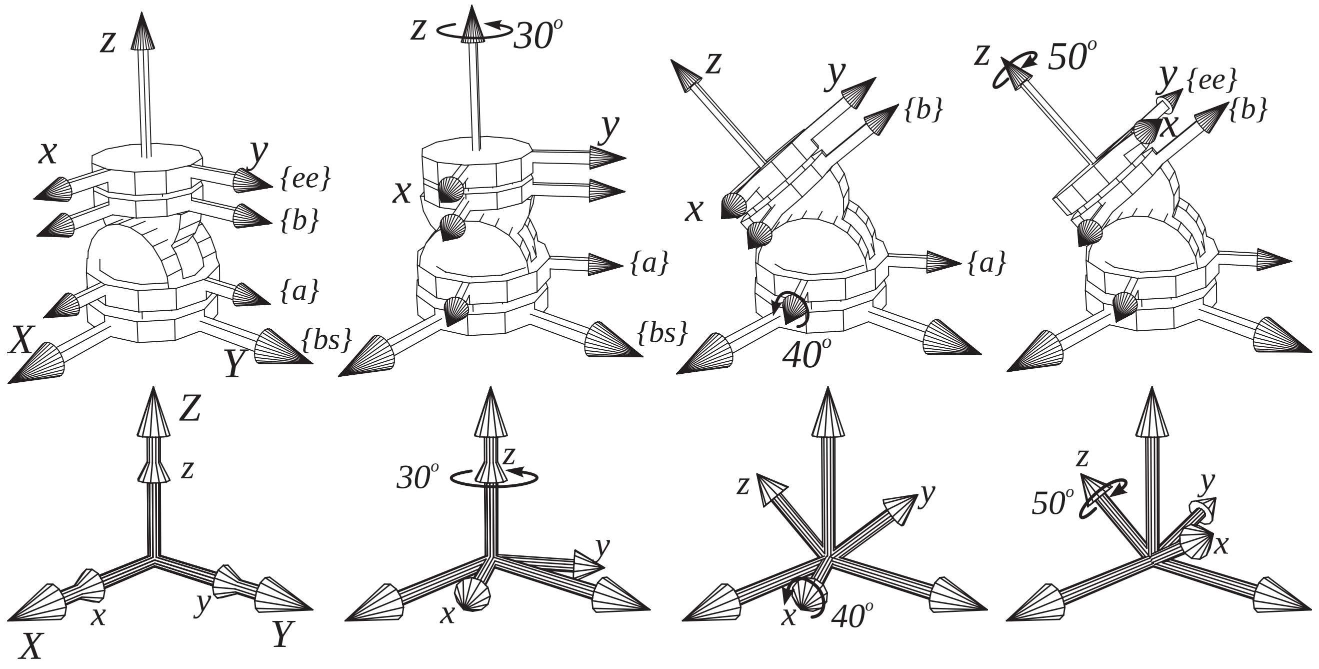


Figure 5.5: Serial kinematic chain ("ZXZ wrist") with three revolute joints, as an example of the composition of rotations about the "moving" z, x and z axes.

3. Finally, its coordinates in frame $\{bs\}$ are ${}^{a}_{bs}\boldsymbol{R}\ {}_{a}\boldsymbol{x}^{ee}$. On the other hand, and by definition, these coordinates are also given by ${}^{ee}_{bs}\boldsymbol{R}\ {}_{ee}\boldsymbol{x}^{ee} = {}^{ee}_{bs}\boldsymbol{R}\ (1\,0\,0)^T$.
4. Similarly for $\boldsymbol{y}^{ee}$ and $\boldsymbol{z}^{ee}$.

Hence

$$\begin{aligned}
{}^{ee}_{bs}\boldsymbol{R} &= {}^{a}_{bs}\boldsymbol{R}\ {}^{b}_{a}\boldsymbol{R}\ {}^{ee}_{b}\boldsymbol{R} \\
&= \boldsymbol{R}(Z,\alpha)\,\boldsymbol{R}(X,\beta)\,\boldsymbol{R}(Z,\gamma) && (5.25)\\
&= \begin{pmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_\beta & -s_\beta \\ 0 & s_\beta & c_\beta \end{pmatrix} \begin{pmatrix} c_\gamma & -s_\gamma & 0 \\ s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} && (5.26)\\
&= \begin{pmatrix} c_\gamma c_\alpha - s_\gamma c_\beta s_\alpha & -s_\gamma c_\alpha - c_\gamma c_\beta s_\alpha & s_\beta s_\alpha \\ c_\gamma s_\alpha + s_\gamma c_\beta c_\alpha & -s_\gamma s_\alpha + c_\gamma c_\beta c_\alpha & -s_\beta c_\alpha \\ s_\gamma s_\beta & c_\gamma s_\beta & c_\beta \end{pmatrix}, && (5.27)
\end{aligned}$$

with the obvious abbreviations $c_\alpha = \cos(\alpha)$, etc. This Eq. (5.27) gives the rotation matrix that corresponds to the ZXZ Euler angles α, β and γ.

Inverse mapping. The mapping $(\alpha, \beta, \gamma) \mapsto \boldsymbol{R}$ must be inverted if one wants to steer a robot wrist as in Figure 5.5 to a desired Cartesian orientation, i.e., a desired orientation $\boldsymbol{R}$ is given, and the corresponding angles

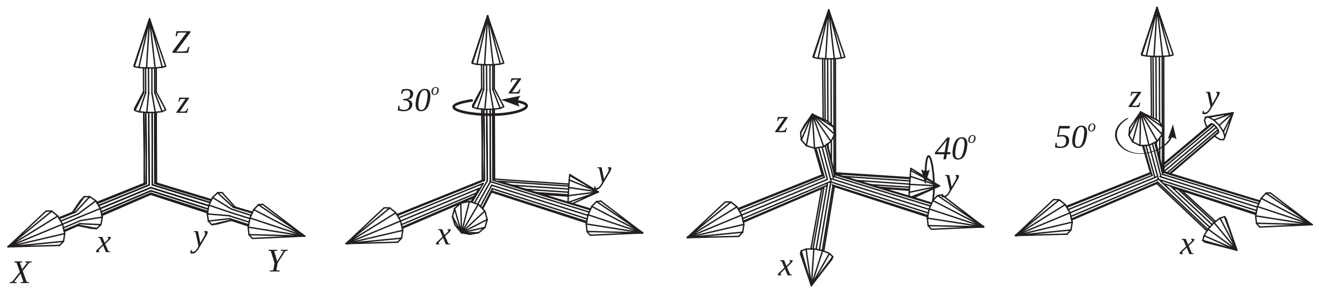


Figure 5.6: The ZYZ Euler angles.

α, β and γ are sought. This inverse can be derived by inspection. α follows from the ratio of $\boldsymbol{R}_{13}(= s_\beta s_\alpha)$ and $\boldsymbol{R}_{23}(= -s_\beta c_\alpha)$:

$$\alpha = \text{atan2}(\boldsymbol{R}_{13}, -\boldsymbol{R}_{23}), \tag{5.28}$$

where "atan2" calculates the arc tangent with the correct quadrant, since it explicitly uses both sine and cosine of the angle and not just their ratio. Then, β is found from the rightmost column in Eq. (5.27):

$$\beta = \text{atan2}(-\boldsymbol{R}_{23} c_\alpha + \boldsymbol{R}_{13} s_\alpha, \boldsymbol{R}_{33}). \tag{5.29}$$

Finally, γ follows from $\boldsymbol{R}_{31}(= s_\gamma s_\beta)$ and $\boldsymbol{R}_{32}(= c_\gamma s_\beta)$:

$$\gamma = \text{atan2}(\boldsymbol{R}_{31}, \boldsymbol{R}_{32}). \tag{5.30}$$

Note the bad numerical conditioning for small β, and the *coordinate singularity* in the inverse relationship if $\beta = 0$: in this case, $\boldsymbol{R}_{13} = \boldsymbol{R}_{23} = \boldsymbol{R}_{31} = \boldsymbol{R}_{32} = 0$. Hence, Eqs (5.28)–(5.30) are not well defined. Physically, this corresponds to situation in which the first and third axes of the wrist in Figure 5.5 are aligned since then $\boldsymbol{R}$ is simply a rotation about the Z axis of $\{bs\}$. It is obvious that, in this aligned situation, this rotation about Z can be achieved by an infinite number of compositions of rotations about the first and third Z axes. But, on the other hand, it is impossible in this situation to apply an angular velocity about the Y-axis of $\{a\}$, i.e., a small rotation about Y needs large rotations about X and Z.

Note also that a *second solution* exists for the inverse calculation (i.e., a different set of Euler angles that gives the same orientation): $c_\beta, s_\beta s_\alpha$ and $-s_\beta c_\alpha$ do not change if β is replaced by $-\beta$, and α by $\alpha + \pi$. In order to keep the last row of the rotation matrix unchanged, γ also has to be replaced by $\gamma + \pi$.

Choice of Euler angles The three-angle sets of Euler angles represent subsequent rotations about axes of a *moving* orthogonal reference frame. The previous paragraphs presented the so-called ZXZ Euler angles. In principle, each triplet of axes gives rise to another set of Euler angles; e.g., Fig. 5.6 shows the ZYZ triplet. However, triplets should not have two identical axes in consecutive places, e.g., ZZX or XYY. Note that no "best" choice exists for the three Euler angles: the appropriateness of a particular set depends on the application. (Euler himself often used different sets, more complicated than the ones that are now named after him, [21].)

The range of the three Euler angles must be limited in order to avoid multiple sets of angles mapping onto the same orientation. For example, in the ZYZ Euler angle representation (Fig. 5.6):

1. The first rotation about Z has a range of $-\pi$ to π. (Inspired by astronomical and geographical terminology, this angle is sometimes called the *azimuth* or *longitude* angle, [7].)

2. The second rotation, about the moved Y axis, has a range of $-\pi/2$ to $\pi/2$. (This angle is called the *elevation* or *latitude* angle, because it determines the "height" above or below the horizon or equator.)

3. The third rotation about Z has a range of $-\pi$ to π. (It is sometimes called the *spin* angle.)

5.3.3 Rotations about fixed axes—Roll, Pitch, Yaw

The obvious next question is now: "What orientation results if one performs $\boldsymbol{R}(Z,\alpha)$, $\boldsymbol{R}(X,\beta)$ and $\boldsymbol{R}(Z,\gamma)$ (in this order) about the axes of the *fixed* reference frame?" The corresponding total orientation is found straightforwardly from the following reasoning using the active rotation interpretation.

1. Start with four coinciding reference frames, $\{bs\} = \{a\} = \{b\} = \{ee\}$.
2. In the first motion, $\{bs\}$ remains in place, and $\{a\}, \{b\}$ and $\{ee\}$ rotate together about the Z-axis of $\{bs\}$ over an angle α. The unit vector $\boldsymbol{x}^{bs}$ along the X axis of the base reference frame is then mapped onto the vector with coordinates ${}_{bs}\boldsymbol{x}^{a} = \boldsymbol{R}(Z,\alpha)\,(1\,0\,0)^T$.
3. $\boldsymbol{R}(X,\beta)$ then rotates frames $\{b\}$ and $\{ee\}$ together about the X-axis of $\{a\}$. This moves the vector ${}_{bs}\boldsymbol{x}^{a}$ further to ${}_{bs}\boldsymbol{x}^{b} = \boldsymbol{R}(X,\beta)\,{}_{bs}\boldsymbol{x}^{a}$.
4. Finally, ${}_{bs}\boldsymbol{x}^{b}$ is moved further to ${}_{bs}\boldsymbol{x}^{ee} = \boldsymbol{R}(Z,\gamma)\,{}_{bs}\boldsymbol{x}^{b}$.

Hence, the total operation is

$$\boldsymbol{R}(ZXZ,\alpha,\beta,\gamma) = \boldsymbol{R}(Z,\gamma)\,\boldsymbol{R}(X,\beta)\,\boldsymbol{R}(Z,\alpha) \tag{5.31}$$

$$= \begin{pmatrix} c_\gamma & -s_\gamma & 0 \\ s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_\beta & -s_\beta \\ 0 & s_\beta & c_\beta \end{pmatrix} \begin{pmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.32}$$

$$= \begin{pmatrix} c_\gamma c_\alpha - s_\gamma c_\beta s_\alpha & -c_\gamma s_\alpha - s_\gamma c_\beta c_\alpha & s_\gamma s_\beta \\ s_\gamma c_\alpha + c_\gamma c_\beta s_\alpha & -s_\gamma s_\alpha + c_\gamma c_\beta c_\alpha & -c_\gamma s_\beta \\ s_\beta s_\alpha & s_\beta c_\alpha & c_\beta \end{pmatrix}. \tag{5.33}$$

Note the difference with Eq. (5.27). Don't try to memorise the order in which rotation matrices are multiplied when composing rotations about fixed or moving frame axes. It's much better to repeat each time the simple reasonings that were used in each of these cases. In operator form, Eqs (5.25) and (5.31) are expressed as

Fact-to-Remember 30 (Moving vs. fixed axed)

$$R^m_{zxz}(\alpha,\beta,\gamma) = \boldsymbol{R}(Z,\alpha)\boldsymbol{R}(X,\beta)\boldsymbol{R}(Z,\gamma), \tag{5.34}$$

and

$$R^f_{zxz}(\alpha,\beta,\gamma) = \boldsymbol{R}(Z,\gamma)\boldsymbol{R}(X,\beta)\boldsymbol{R}(Z,\alpha). \tag{5.35}$$

The superscripts "m" and "f" indicate that the rotations take place about moving, respectively fixed frame axes. The subscript denotes the order of the rotations. The parameter values are the corresponding rotation angles.

Roll-Pitch-Yaw angles. If the rotation angles are small, the Euler angle sets with common first and third rotation axes (e.g., *ZYZ* or *ZXZ*) are badly conditioned numerically, since the spatial directions of these first and third axes differ only slightly. (Recall the problems with small β in Eqs (5.28)–(5.30).) For many centuries already, this situation has been very common for sea navigation, and hence, *Roll-Pitch-Yaw* have been introduced, describing rotations about three *orthogonal* axes *fixed* to the moving object or vehicle. This name still reminds its nautical origin. The Roll-Pitch-Yaw angles represent the orientation of a frame, by subsequent rotations about the vertical (Z, "yaw" y), transverse (Y, "pitch" p) and longitudinal (X, "roll" r) axes of the moving rigid body (Fig. 5.7). The ZYX Euler angles are introduced here as rotations about *moving* axes, but as shown in the previous subsection they are equivalent to rotations about, respectively, the *fixed* X, Y and Z axes, over the same angles (Fig. 5.8). Hence, the rotation matrix corresponding to the ZYX or Roll-Pitch-Yaw Euler angles is

$$\begin{aligned}
\boldsymbol{R}(RPY, r, p, y) &= \boldsymbol{R}(ZYX, y, p, r) \\
&= \boldsymbol{R}(Z, y)\,\boldsymbol{R}(Y, p)\,\boldsymbol{R}(X, r) \\
&= \begin{pmatrix} c_y & -s_y & 0 \\ s_y & c_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_p & 0 & s_p \\ 0 & 1 & 0 \\ -s_p & 0 & c_p \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_r & -s_r \\ 0 & s_r & c_r \end{pmatrix} \\
&= \begin{pmatrix} c_y c_p & c_y s_p s_r - s_y c_r & c_y s_p c_r + s_y s_r \\ s_y c_p & s_y s_p s_r + c_y c_r & s_y s_p c_r - c_y s_r \\ -s_p & c_p s_r & c_p c_r \end{pmatrix}.
\end{aligned} \tag{5.36}$$

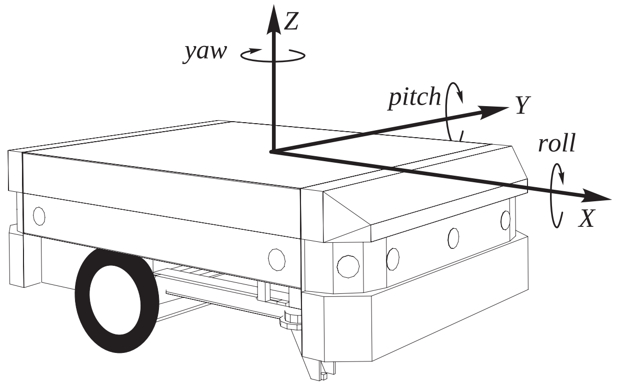


Figure 5.7: Roll-Pitch-Yaw angles for a mobile robot. (Figure courtesy of W. Persoons.)

Inverse of RPY. The *inverse* relationship calculates roll r, pitch p and yaw y from a given rotation matrix $\boldsymbol{R}$. As for the ZXZ Euler angles, these relationships are easily derived by inspection of Eq. (5.36); for example,

$$r = \text{atan2}(\boldsymbol{R}_{32}, \boldsymbol{R}_{33}), \tag{5.37}$$

$$y = \text{atan2}(\boldsymbol{R}_{21}, \boldsymbol{R}_{11}), \tag{5.38}$$

$$p = \text{atan2}(-\boldsymbol{R}_{31}, c_y \boldsymbol{R}_{11} + s_y \boldsymbol{R}_{21}). \tag{5.39}$$

Note some similarities with the Euler angles of Eqs (5.28)–(5.30):

1. The equations above are badly conditioned numerically if $c_p \approx 0$. This case corresponds to $p \approx \pi/2$ or $-\pi/2$, i.e., a "large" angle; but, as mentioned above, the Roll-Pitch-Yaw Euler angles have been introduced historically for small angles only.

2. A second solution is found by replacing p by $\pi - p$, r by $r + \pi$ and y by $y + \pi$.

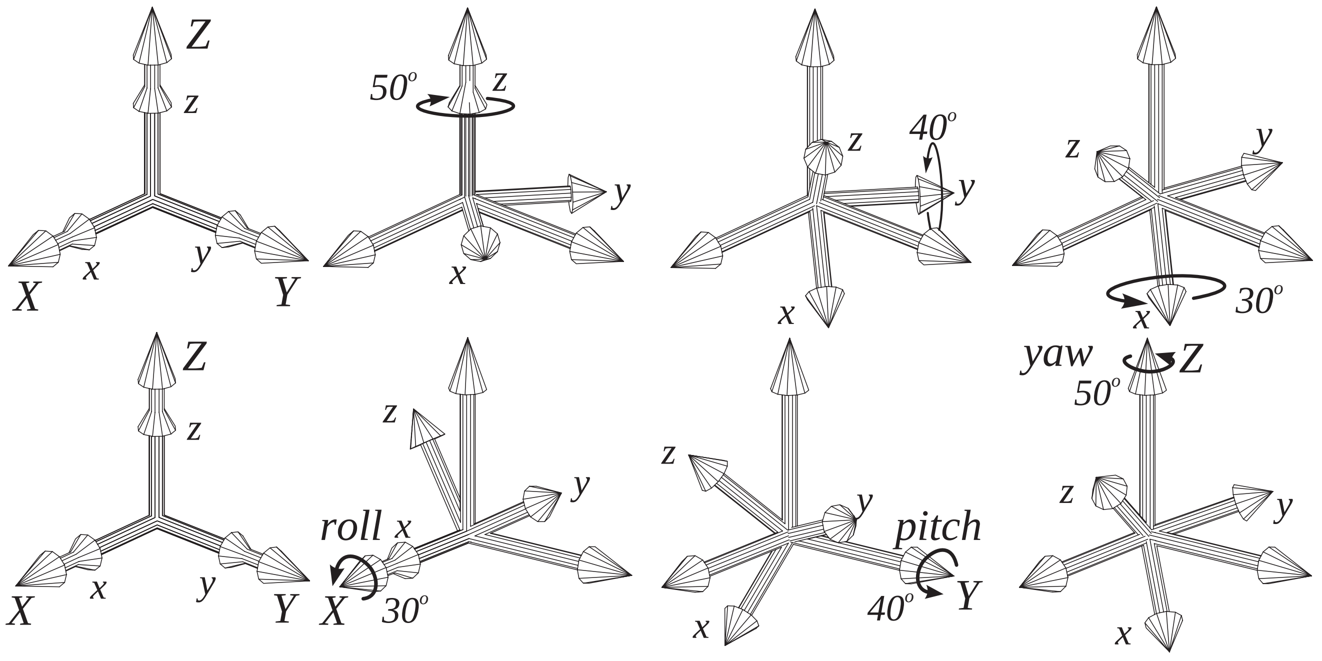


Figure 5.8: *ZYX* Euler angles (top) and Roll-Pitch-Yaw angles (bottom), both corresponding to the same orientation.

5.3.4 Advantages and disadvantages

Three-angle orientation representations have two advantages:

1. They use the *minimal* number of parameters.
2. One can choose a set of three angles that, by design, feels "natural" or "intuitive" for a given application. For example, the orientations of the specific robotic wrist in Figure 5.5 are naturally represented by *ZXZ* Euler angles. Or, the roll, pitch and yaw motions of a ship or airplane definitely live up to their names in rough weather.

However, it is a (not so well-known)

Fact-to-Remember 31 (Singularities of Euler angles)
No set of three angles can globally represent all orientations without singularity, [12].

This means that a set of neighbouring orientations cannot always be represented by a set of neighbouring Euler angles. (The converse *is* true: the rotation matrix in, for example, Eq. (5.36) is a continuous function of its Euler angle parameters.) For example, the robot wrist in Figure 5.5 has a singularity when two axes line up. This happens when the rotation about the second axis brings the third axis parallel to the first. Indeed, two orientations nearly parallel to the base frame of the wrist, but with their origins lying on opposite sides of the Y axis, cannot be given Euler angle values that lie close to each other. This can cause problems if the robot controller blindly interpolates between the initial orientation and the desired end orientation. Another example: a small rotation about the X-axis requires large rotations of the joints in a ZYZ wrist.

Inverse relations such as Eqs (5.38)-(5.39) always become singular for some particular values of the Euler angles. Physically, this corresponds to the fact that *any* spherical wrist with three joints has configurations in which two of the axes line up.

Some orientations don't have *unique* Euler angles. For example, the ZXZ Euler angle set described above is not one-to-one if the angle ranges include the limits $\pm\pi$ or $\pm\pi/2$: the "north" and "south poles" are covered an infinite number of times. Finally, one should be aware of this

Fact-to-Remember 32 (Euler angles are not a vector)
No set of three Euler angles is a vector:
1. Adding two sets of Euler angles does not give the set of Euler angles that corresponds to the composed orientation.
2. The order of rotations matters, i.e., composition of rotations is not commutative, while vector addition is.

5.3.5 Euler angle time rates and angular velocity

Equation (5.20) represents the relation between the time rate of an orientation matrix and the instantaneous angular velocity of the moving frame. This paragraph deduces a similar relationship between the angular velocity $\boldsymbol{\omega}$ and the time derivatives of the Euler angles.

Take again the example of the ZXZ Euler angles rotation of Figure 5.5. In an orientation with given α, β and γ, the time rate of the angle α generates an instantaneous angular velocity (with magnitude $\dot{\alpha}$) about the Z axis of the fixed frame. The time rate of the angle β gives an instantaneous angular velocity (with magnitude $\dot{\beta}$) about the X axis that has been moved by $\boldsymbol{R}(Z,\alpha)$, and hence is currently pointing in the direction ${}^{a}_{bs}\boldsymbol{R}\,(1\;0\;0)^T = (c_\alpha\; s_\alpha\; 0)^T$. The time rate of the angle γ has a magnitude $\dot{\gamma}$, and takes place about the Z axis of the frame moved first by $\boldsymbol{R}(Z,\alpha)$ and then by $\boldsymbol{R}(X,\beta)$, and hence is pointing in the direction ${}^{a}_{bs}\boldsymbol{R}\;{}^{b}_{a}\boldsymbol{R}\,(0\;0\;1)^T = (s_\beta s_\alpha\;\; -s_\beta c_\alpha\;\; c_\beta)^T$. Summing these three angular velocity contributions, the total angular velocity three-vector $\boldsymbol{\omega}$ is

$$\boxed{\begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix} = \begin{pmatrix}0 & c_\alpha & s_\beta s_\alpha\\ 0 & s_\alpha & -s_\beta c_\alpha\\ 1 & 0 & c_\beta\end{pmatrix}\begin{pmatrix}\dot{\alpha}\\ \dot{\beta}\\ \dot{\gamma}\end{pmatrix}.} \tag{5.40}$$

Inverse relationship. Some simple algebra yields the *inverse* of this relationship:

$$\boxed{\begin{pmatrix}\dot{\alpha}\\ \dot{\beta}\\ \dot{\gamma}\end{pmatrix} = \begin{pmatrix}-\dfrac{s_\alpha c_\beta}{s_\beta} & \dfrac{c_\alpha c_\beta}{s_\beta} & 1\\ c_\alpha & s_\alpha & 0\\ \dfrac{s_\alpha}{s_\beta} & -\dfrac{c_\alpha}{s_\beta} & 0\end{pmatrix}\begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix}.} \tag{5.41}$$

In correspondence to the singularity analysis above, this inverse becomes singular for $\beta = 0$, i.e., when the first and third joint axes of the spherical wrist in Figure 5.5 line up. In this configuration, no angular velocity about the Y axis is possible.

5.3.6 Integrability of angular velocity

The angular *velocity* is represented by a three-vector $\boldsymbol{\omega}$; a three-vector of Euler angles represents the *orientation.* Integrating the angular velocity over a certain amount of time results in a change of Euler angles. But:

Fact-to-Remember 33 (Angular velocity and Euler angle derivatives)
The angular velocity three-vector $\boldsymbol{\omega}$ is <u>not exact</u>, i.e., it is not the time derivative of any Euler angle set, [11, p. 347]. However, <u>integrating factors</u> such as in Eq. (5.40) exist.

The non-exactness is proved as follows. Consider, for example, the Y component of $\boldsymbol{\omega}$ in Eq. (5.40). In "differential" form, this gives

$$s_\alpha d\beta - s_\beta c_\alpha d\gamma \triangleq A(\alpha,\beta,\gamma)d\alpha + B(\alpha,\beta,\gamma)d\beta + C(\alpha,\beta,\gamma)d\gamma. \tag{5.42}$$

Such a differential form is integrable (or *exact* [3, 22, 24, 27]) if and only if

$$\frac{\partial A}{\partial \beta} = \frac{\partial B}{\partial \alpha}, \tag{5.43}$$

as well as all similar combinations. That this condition is not satisfied is easily checked from Eq. (5.42):

$$\frac{\partial C}{\partial \beta} = -c_\beta c_\alpha, \quad \text{but} \quad \frac{\partial B}{\partial \gamma} = 0. \tag{5.44}$$

5.4 Equivalent axis and equivalent angle of rotation

Euler's Theorem (Fact 29) says that any displacement of a rigid body in which (at least) one point remains fixed, is a rotation about some axis. In other words, every rotation matrix $\boldsymbol{R}$ is generated by one single rotation, about a certain axis represented by the unit vector $\boldsymbol{e}^{eq}$, and over a certain angle θ. $\boldsymbol{e}^{eq}$ is the unit vector along the so-called *equivalent rotation axis*, and θ is the *equivalent rotation angle.* The obvious question, of course, is how to find these equivalent parameters from the rotation matrix, and vice versa?

5.4.1 Forward relation

If the axis $\boldsymbol{e}^{eq} = (\boldsymbol{e}^{eq}_x \; \boldsymbol{e}^{eq}_y \; \boldsymbol{e}^{eq}_z)^T$ and the angle θ are known, the corresponding rotation matrix $\boldsymbol{R}(\boldsymbol{e}^{eq},\theta)$ is found as a sequence of five *frame axis rotations* about *fixed* axes (Fig. 5.9):

1. Rotate the equivalent axis about the Z axis until it lies in the XZ plane. This is done by rotation matrix $\boldsymbol{R}(Z,\alpha)$, with $\alpha = -\arctan(\boldsymbol{e}^{eq}_y/\boldsymbol{e}^{eq}_x)$.
2. Rotate this new axis about the Y axis until it coincides with the X axis. This is done by rotation matrix $\boldsymbol{R}(Y,\beta)$, with $\beta = \arctan\left(\boldsymbol{e}^{eq}_z/\left((\boldsymbol{e}^{eq}_x)^2 + (\boldsymbol{e}^{eq}_y)^2\right)\right)$.
3. Perform the rotation about the angle θ: $\boldsymbol{R}(X,\theta)$.
4. Execute the first two rotations in reverse order, i.e., bring the equivalent axis back to its original position. Hence

$$\boldsymbol{R}(\boldsymbol{e}^{eq},\theta) = \boldsymbol{R}(Z,-\alpha)\boldsymbol{R}(Y,-\beta)\boldsymbol{R}(X,\theta)\boldsymbol{R}(Y,\beta)\boldsymbol{R}(Z,\alpha), \tag{5.45}$$

or

$$\boxed{\boldsymbol{R}(\boldsymbol{e}^{eq},\theta) = \begin{pmatrix} (\boldsymbol{e}^{eq}_x)^2 v_\theta + c_\theta & \boldsymbol{e}^{eq}_x\boldsymbol{e}^{eq}_y v_\theta - \boldsymbol{e}^{eq}_z s_\theta & \boldsymbol{e}^{eq}_x\boldsymbol{e}^{eq}_z v_\theta + \boldsymbol{e}^{eq}_y s_\theta \\ \boldsymbol{e}^{eq}_x\boldsymbol{e}^{eq}_y v_\theta + \boldsymbol{e}^{eq}_z s_\theta & (\boldsymbol{e}^{eq}_y)^2 v_\theta + c_\theta & \boldsymbol{e}^{eq}_y\boldsymbol{e}^{eq}_z v_\theta - \boldsymbol{e}^{eq}_x s_\theta \\ \boldsymbol{e}^{eq}_x\boldsymbol{e}^{eq}_z v_\theta - \boldsymbol{e}^{eq}_y s_\theta & \boldsymbol{e}^{eq}_y\boldsymbol{e}^{eq}_z v_\theta + \boldsymbol{e}^{eq}_x s_\theta & (\boldsymbol{e}^{eq}_z)^2 v_\theta + c_\theta \end{pmatrix}.} \tag{5.46}$$

c_θ and s_θ are shorthand notations for $\cos(\theta)$ and $\sin(\theta)$, respectively. v_θ is the "verse of theta," which is equal to $1 - c_\theta$.

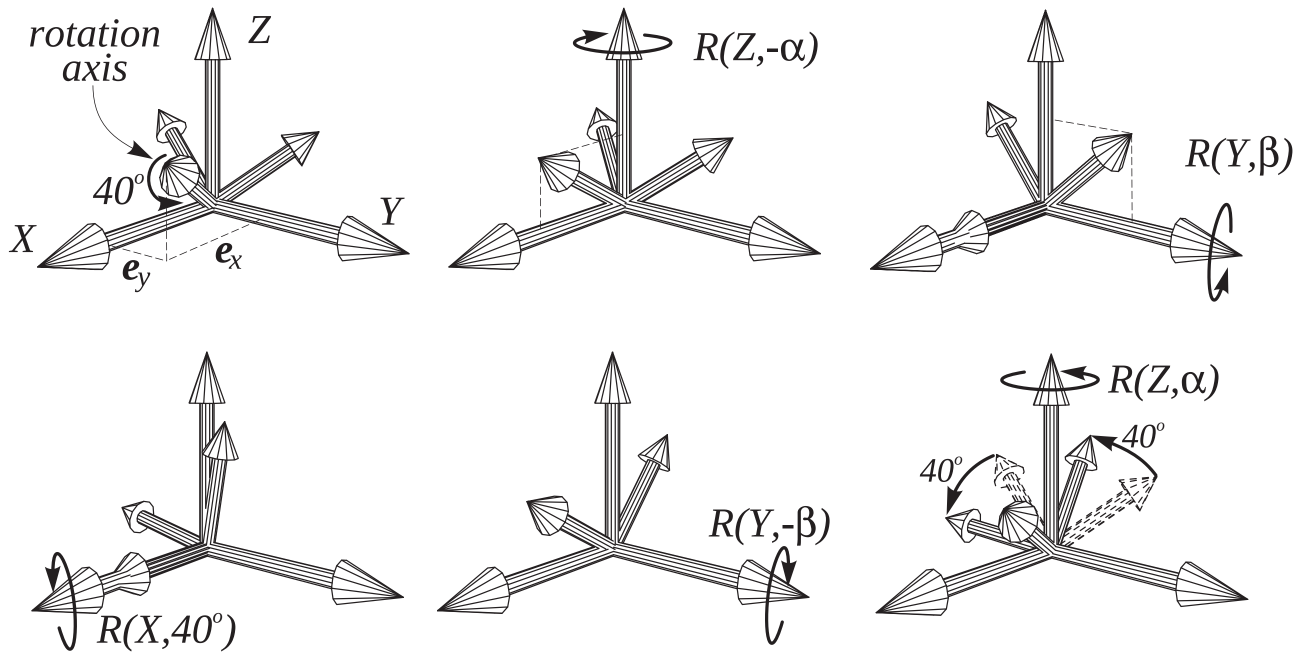


Figure 5.9: Rotation about an arbitrary axis is equivalent to a sequence of five rotations about the fixed axes.

5.4.2 Inverse relation

The transformations from rotation matrix to equivalent axis parameters are deduced from Eq. (5.46), via the following observations, [5, 7, 11, 16]:

1. The sum of the diagonal elements of $\boldsymbol{R}(\boldsymbol{e}^{eq},\theta)$ (i.e., its *trace*) is

$$\text{trace}\left(\boldsymbol{R}(\boldsymbol{e}^{eq},\theta)\right) = 1 + 2c_\theta. \tag{5.47}$$

Hence

$$\boxed{\theta = \arccos\left(\frac{\text{trace}\left(\boldsymbol{R}(\boldsymbol{e}^{eq},\theta)\right) - 1}{2}\right).} \tag{5.48}$$

This inverse has two solutions; the second one is found from the first by rotating in the other sense of the equivalent axis, and over the negative equivalent angle. The equivalent rotation angle can also be considered as the magnitude of the angular velocity about the equivalent axis that yields the given rotation matrix if applied during one unit of time.

2. Subtracting pairs of off-diagonal terms gives

$$\begin{array}{rcl} \boldsymbol{R}_{32}(\boldsymbol{e}^{eq},\theta) - \boldsymbol{R}_{23}(\boldsymbol{e}^{eq},\theta) & = & 2\boldsymbol{e}^{eq}_{x} s_\theta, \\ \boldsymbol{R}_{13}(\boldsymbol{e}^{eq},\theta) - \boldsymbol{R}_{31}(\boldsymbol{e}^{eq},\theta) & = & 2\boldsymbol{e}^{eq}_{y} s_\theta, \\ \boldsymbol{R}_{21}(\boldsymbol{e}^{eq},\theta) - \boldsymbol{R}_{12}(\boldsymbol{e}^{eq},\theta) & = & 2\boldsymbol{e}^{eq}_{z} s_\theta. \end{array} \tag{5.49}$$

These equations cannot be inverted for $\theta = 0$ or $\theta = \pi$. However, these cases are trivially recognized from the rotation matrix. If $\theta \notin \{0, \pi\}$, the equivalent axis unit vector is

$$\boxed{\boldsymbol{e}^{eq} = \frac{1}{2s_\theta}\begin{pmatrix} \boldsymbol{R}_{32}(\boldsymbol{e}^{eq},\theta) - \boldsymbol{R}_{23}(\boldsymbol{e}^{eq},\theta) \\ \boldsymbol{R}_{13}(\boldsymbol{e}^{eq},\theta) - \boldsymbol{R}_{31}(\boldsymbol{e}^{eq},\theta) \\ \boldsymbol{R}_{21}(\boldsymbol{e}^{eq},\theta) - \boldsymbol{R}_{12}(\boldsymbol{e}^{eq},\theta) \end{pmatrix}.} \tag{5.50}$$

Note that these equations are numerically not very well conditioned for $\theta \approx 0$ and $\theta \approx \pi$!

Logarithm. The procedure above also produces the "*logarithm*" of a rotation matrix, i.e., the angular velocity that generates the given rotation matrix in one unit of time. Recall that the exponential maps elements from the "tangent space" (i.e., velocities) to the manifold; the logarithm is a mapping in the opposite sence.

5.4.3 Time derivative

Equation (5.45) gives the relationship between a rotation about the arbitrary axis along $\boldsymbol{e}^{eq}$ and the same rotation about the X-axis of the inertial frame. Using Eq. (5.20) for the time derivative of a rotation matrix yields the relationship between the corresponding angular velocities $\boldsymbol{\omega}^{eq}$ and $\boldsymbol{\omega}_x$ about both axes:

$$\begin{aligned} [\boldsymbol{\omega}^{eq}] &= \dot{\boldsymbol{R}}(\boldsymbol{e}^{eq},\theta)\ \boldsymbol{R}^{-1}(\boldsymbol{e}^{eq},\theta) \\ &= \Big\{\boldsymbol{R}(Z,-\alpha)\boldsymbol{R}(Y,-\beta)\dot{\boldsymbol{R}}(X,\theta)\boldsymbol{R}(Y,\beta)\boldsymbol{R}(Z,\alpha)\Big\}\Big\{\boldsymbol{R}(Z,-\alpha)\boldsymbol{R}(Y,-\beta)\boldsymbol{R}(X,\theta)\boldsymbol{R}(Y,\beta)\boldsymbol{R}(Z,\alpha)\Big\} \\ &= \boldsymbol{R}(Z,-\alpha)\boldsymbol{R}(Y,-\beta)\,[\boldsymbol{\omega}_x]\,\boldsymbol{R}(Y,\beta)\boldsymbol{R}(Z,\alpha). \end{aligned} \tag{5.51}$$

5.4.4 Similarity transformations

The procedure applied in deriving Eq. (5.45) works for general transformations too, not just rotations about frame axes. So, the following Chapters of this book will often use these

Fact-to-Remember 34 (Similarity transformations)
Often, a general transformation T can be written as

$$T = S^{-1}T'S, \tag{5.52}$$

where S and T' are (sequences) of <u>*elementary*</u> *(invertible) transformations; T' is of the same "type" as T. In the section above, the elementary transformations are rotations about* <u>*frame*</u> *axes.*

Exponential of general angular velocity. A first example of this similarity transformation principle is the exponential of an angular velocity about an *arbitrary* axis with direction vector $\boldsymbol{e}^{eq}$. Section 5.2.10 proved that any rotation about one of the *frame* axes corresponds to the exponentiation of an angular velocity about this axis, Eq. (5.21). According to Eq. (5.45), a rotation $\boldsymbol{R}(\boldsymbol{e}^{eq}, \theta)$ about the axis $\boldsymbol{e}^{eq}$ over the angle θ can be written in the form of Eq. (5.52), with the matrix $\boldsymbol{S}$ (corresponding to the transformation S) equal to $\boldsymbol{R}(Y, \beta)\boldsymbol{R}(Z, \alpha)$ and, similarly, $\boldsymbol{T}' = \boldsymbol{R}(X, \theta) = \exp(\boldsymbol{A})$, with $\boldsymbol{A} = [\boldsymbol{\omega}_x]$ the skew-symmetric matrix corresponding to the angular velocity that makes the X axis rotate over the angle θ in one unit of time. Equation (5.22) and (5.51) then imply that

$$\begin{aligned}
\boldsymbol{R}(\boldsymbol{e}^{eq}, \theta) &= \boldsymbol{S}^{-1} \exp(\boldsymbol{A}) \boldsymbol{S} \\
&= \boldsymbol{S}^{-1} \left(\mathbf{1} + \boldsymbol{A} + \frac{\boldsymbol{A}^2}{2!} + \dots \right) \boldsymbol{S} \\
&= \mathbf{1} + \left(\boldsymbol{S}^{-1} \boldsymbol{A} \boldsymbol{S} \right) + \frac{\boldsymbol{S}^{-1} \boldsymbol{A} \boldsymbol{S} \boldsymbol{S}^{-1} \boldsymbol{A} \boldsymbol{S}}{2!} + \dots \\
&= \exp \left(\boldsymbol{S}^{-1} \boldsymbol{A} \boldsymbol{S} \right) \\
&= \exp \left([\boldsymbol{\omega}^{eq}] \right). \qquad (5.53)
\end{aligned}$$

$\boldsymbol{\omega}^{eq}$ is the angular velocity about the initial arbitrary axis $\boldsymbol{e}^{eq}$ that generates the rotation over an angle θ in one unit of time. Hence, the exponential formula is valid for angular velocities about arbitrary axes.

5.4.5 Distance between two orientations

The distance between two orientations (and hence, the distance between the two corresponding rotation matrices, $\boldsymbol{R}^1$ and $\boldsymbol{R}^2$) can be defined independently of the chosen representation, [16, 19]. (Hence, it is a *structural* property of relative orientations.) First, take the relative orientation $\boldsymbol{R} = \left(\boldsymbol{R}^1\right)^{-1} \boldsymbol{R}^2$. As described in the previous paragraphs, $\boldsymbol{R}$ corresponds to a rotation about an equivalent axis $\boldsymbol{e}^{eq}$, over an angle θ. Now,

Fact-to-Remember 35 (Logarithm is distance function on *SO(3)*)
The distance between two orientations $\boldsymbol{R}^1$ and $\boldsymbol{R}^2$ is the equivalent angle of rotation (or the logarithm) of the relative orientation $\boldsymbol{R} = \left(\boldsymbol{R}^1\right)^{-1} \boldsymbol{R}^2$. It is the magnitude of the angular velocity that can close the orientation gap in one unit of time.

It can be proved that this rotation angle is *smaller* than the sum of any set of angles used in other orientation representations, such as for example Euler angle sets. Note the similarity of this property to the case of the Euclidean distance between points, with (i) the composition of rotations (i.e., matrix multiplication in the case of rotation matrix representation) replaced by composition of position (i.e., addition of vectors) and (ii) the inverse replaced by the negative.

5.5 Unit quaternions

The previous Sections presented rotation matrices (that have no coordinate singularities, but use much more parameters than strictly necessary), and Euler angle sets (that suffer from coordinate singularities, but use only the minimal number of parameters). This Section presents yet another representation, that has become popular because it is an interesting compromise between the advantages and disadvantages of both other representations.

5.5.1 Definition and use

Another interesting orientation representation is the *four*-parameter set of *unit quaternions*, also called *Euler-(Rodrigues) parameters*, [4, 26]:

Fact-to-Remember 36 (Unit quaternions)
If the equivalent axis $\boldsymbol{e}^{eq}$ of a rotation is known, as well as the corresponding equivalent rotation angle θ, then the unit quaternion $\boldsymbol{q}$ representing the same rotation is defined as the following four-vector:

$$\boldsymbol{q}(\boldsymbol{e}^{eq}, \theta) \triangleq \begin{pmatrix} \boldsymbol{q}_v \\ q \end{pmatrix} = \begin{pmatrix} s_{\frac{\theta}{2}}\ \boldsymbol{e}^{eq}_x \\ s_{\frac{\theta}{2}}\ \boldsymbol{e}^{eq}_y \\ s_{\frac{\theta}{2}}\ \boldsymbol{e}^{eq}_z \\ c_{\frac{\theta}{2}} \end{pmatrix}. \tag{5.54}$$

Since, in general, two equivalent rotation angles and axes exist for every rotation (Sect. 5.4.2), there also exist two quaternions for each single orientation: $(\boldsymbol{q}_v^T, q)^T$ and $(-\boldsymbol{q}_v^T, q)^T$.

$\boldsymbol{q}_v$ is the *vector part* of the unit quaternion $\boldsymbol{q}$; q is the scalar part. (Some other references interchange the order of the vector and scalar parts, but this has no physical meaning nor consequences.) $\boldsymbol{q}$ is called a *unit quaternion* because it has "Euclidean" unit two-norm:

$$\boxed{\boldsymbol{q}^T\boldsymbol{q} = \boldsymbol{q}_v^T\boldsymbol{q}_v + q^2 = 1.} \tag{5.55}$$

Unit quaternions have become quite popular in the robotics community only recently, although the Irish mathematician Sir William Rowan Hamilton (1805–1865) described the quaternions already more than a century ago. Hamilton indeed wrote a very impressive pair of books about the subject of quaternions [10] even *before* the dot product and cross product between three-vectors were introduced by Josiah Willard Gibbs (1839–1903), [6]. The parameterization of rotations by means of quaternions was already described by Olinde Rodrigues in 1840 [8, 23, 29], and even earlier by Johann Carl Friedrich Gauss (1777–1855), who didn't bother to publish about them. Hamilton's original goal was to come up with a generalization of the complex numbers: while the set of complex numbers is generated by the two numbers 1 and $i \triangleq \sqrt{-1}$, the quaternions have four generating four-vectors $\boldsymbol{1}, \boldsymbol{i}, \boldsymbol{j}$ and $\boldsymbol{k}$:

$$\boldsymbol{i} \triangleq \boldsymbol{q}(\boldsymbol{e}_x, \pi) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \boldsymbol{j} \triangleq \boldsymbol{q}(\boldsymbol{e}_y, \pi) = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \boldsymbol{k} \triangleq \boldsymbol{q}(\boldsymbol{e}_z, \pi) = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \boldsymbol{1} \triangleq \boldsymbol{q}(\times, 0) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{5.56}$$

So, $\boldsymbol{i}, \boldsymbol{j}$ and $\boldsymbol{k}$ correspond to rotations over an angle π about the X, Y and Z axes, respectively; $\boldsymbol{1}$ corresponds to the unit rotation matrix. These four generators have algebraic properties that are a generalization of the complex number generators 1 and i:

$$\boxed{\boldsymbol{i}\boldsymbol{j} = \boldsymbol{k}, \quad \boldsymbol{i}\boldsymbol{j} = -\boldsymbol{j}\boldsymbol{i}, \quad \boldsymbol{i}\boldsymbol{i} = -1,} \tag{5.57}$$

plus all cyclic permutations.

Why quaternions? The following fact states the most prominent reason to use quaternions in robotics:

Fact-to-Remember 37 (Singularity free)
A unit quaternion is the orientation representation with the smallest number of parameters that can represent all orientations without singularity, [1, 28]. This means that one can move between any two orientations in a smooth way, i.e., with only smooth changes in the quaternion parameters.

This fact has important consequences for *trajectory generation* (also called *path planning*): without knowing in advance the initial and final orientations of the robot, one will never encounter a singularity when using a quaternion representation to interpolate between these two orientations. (Recall that this is not true for Euler angle interpolation.) Moreover, since we defined quaternions in Eq. (5.54) in terms of the equivalent axis and the equivalent rotation angle, *interpolation* of orientations by means of quaternions boils down to interpolating the equivalent rotation angle about the equivalent axis. Besides these definite advantages, one does have to keep in mind the following "problems":

1. The two-to-one orientation representation, Fact 36. Hence, it is important to choose the correct sign during continuous interpolation problems, since all switches between the two alternatives would cause jumps in the generated trajectory in "quaternion configuration space."
2. Bad numeric conditioning (for equivalent rotation angles of about 0 or π) of the problem of extracting the equivalent axis from a rotation matrix, Eq. (5.50).

5.5.2 Multiplication of quaternions

Rotations correspond to a somewhat unusual multiplication of quaternions. This Section presents quaternion multiplication; the next Section will make the link with rotation matrices.

Two quaternions $\boldsymbol{q}^1 = x^1\boldsymbol{i} + y^1\boldsymbol{j} + z^1\boldsymbol{k} + s^1$ and $\boldsymbol{q}^2 = x^2\boldsymbol{i} + y^2\boldsymbol{j} + z^2\boldsymbol{k} + s^2$ are multiplied according to the algebraic rules in Eq. (5.57):

$$\boxed{\boldsymbol{q}^1\boldsymbol{q}^2 = \begin{pmatrix} y^1z^2 - y^2z^1 + x^1s^2 + x^2s^1 \\ z^1x^2 - z^2x^1 + y^1s^2 + y^2s^1 \\ x^1y^2 - x^2y^1 + z^1s^2 + z^2s^1 \\ s^1s^2 - x^1x^2 - y^1y^2 - z^1z^2 \end{pmatrix}}. \tag{5.58}$$

(See e.g., [7, 10, 11, 15, 17, 18] for more detailed algebraic discussions.) The right-hand side can be re-organised in a scalar part and a vector part:

$$\boldsymbol{q}^1\boldsymbol{q}^2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ s^1s^2 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 0 \\ x^1x^2 + y^1y^2 + z^1z^2 \end{pmatrix} + s^1 \begin{pmatrix} x^2 \\ y^2 \\ z^2 \\ 0 \end{pmatrix} + s^2 \begin{pmatrix} x^1 \\ y^1 \\ z^1 \\ 0 \end{pmatrix} + \begin{pmatrix} y^1z^2 - y^2z^1 \\ z^1x^2 - z^2x^1 \\ x^1y^2 - x^2y^1 \\ 0 \end{pmatrix} \tag{5.59}$$

$$= \left(q^1q^2 - \boldsymbol{q}_v^1 \cdot \boldsymbol{q}_v^2\right) + \left(q^1\boldsymbol{q}_v^2 + q^2\boldsymbol{q}_v^1 + \boldsymbol{q}_v^1 \times \boldsymbol{q}_v^2\right). \tag{5.60}$$

This shows more clearly that the quaternion product of four-vectors is a generalization of the more familiar dot product (second term) and cross product (last term) of three-vectors. The quaternion product can also be represented by a *matrix multiplication*, [13, 15] (which is alway handy to *compose* operations):

$$\boxed{\boldsymbol{q}^1\boldsymbol{q}^2 = \boldsymbol{Q}_l^1\boldsymbol{q}^2 = \boldsymbol{Q}_r^2\boldsymbol{q}^1,} \tag{5.61}$$

with

$$\boldsymbol{Q}_l^1 \triangleq \begin{pmatrix} s^1 & -z^1 & y^1 & x^1 \\ z^1 & s^1 & -x^1 & y^1 \\ -y^1 & x^1 & s^1 & z^1 \\ -x^1 & -y^1 & -z^1 & s^1 \end{pmatrix}, \qquad \boldsymbol{Q}_r^2 \triangleq \begin{pmatrix} s^2 & z^2 & -y^2 & x^2 \\ -z^2 & s^2 & x^2 & y^2 \\ y^2 & -x^2 & s^2 & z^2 \\ -x^2 & -y^2 & -z^2 & s^2 \end{pmatrix}. \tag{5.62}$$

$\boldsymbol{Q}^i$ denotes the matrix corresponding to the quaternion $\boldsymbol{q}^i$. The subscripts "l" and "r" indicate whether the quaternion corresponding to the matrix multiplies on the left or on the right, respectively. Both "l" and "r" matrices are *orthogonal*, since the corresponding quaternions are *unit quaternions*. Hence, $\boldsymbol{q}^{-1} = (-\boldsymbol{q}_v \ q)^T$ is the *inverse* of $\boldsymbol{q} = (\boldsymbol{q}_v \ q)^T$, and

$$\boldsymbol{Q}(\boldsymbol{q}^{-1}) = \boldsymbol{Q}^T(\boldsymbol{q}), \quad \boldsymbol{Q}_r^{-1} = \boldsymbol{Q}_r^T, \quad \boldsymbol{Q}_l^{-1} = \boldsymbol{Q}_l^T. \tag{5.63}$$

5.5.3 Unit quaternions and rotations

We now have sufficient algebraic definitions to describe how a quaternion operates on a three-vector to execute a rotation. First, re-define *formally* a three-vector $\boldsymbol{p}$ as a four-vector quaternion $\boldsymbol{p} = (\boldsymbol{p} \ 0)^T$. (Note the obvious abuse of notation!) Then, the transformation of $\boldsymbol{p}$ into $\boldsymbol{p\prime}$ by the rotation represented by $\boldsymbol{q}$ is given by, [11, 15],

$$\boxed{\boldsymbol{p\prime} = \boldsymbol{q}\boldsymbol{p}\boldsymbol{q}^{-1}, \qquad \boldsymbol{p\prime} = \boldsymbol{Q}_l\boldsymbol{Q}_r^T\boldsymbol{p}.} \tag{5.64}$$

Indeed, for $\boldsymbol{q} = (s_{\frac{\theta}{2}}\boldsymbol{e}^T \ c_{\frac{\theta}{2}})^T$, $\boldsymbol{Q}_l$ and $\boldsymbol{Q}_r$ are given by

$$\boldsymbol{Q}_l = \begin{pmatrix} c_{\frac{\theta}{2}} & -s_{\frac{\theta}{2}}\ \boldsymbol{e}_z & s_{\frac{\theta}{2}}\ \boldsymbol{e}_y & s_{\frac{\theta}{2}}\ \boldsymbol{e}_x \\ s_{\frac{\theta}{2}}\ \boldsymbol{e}_z & c_{\frac{\theta}{2}} & -s_{\frac{\theta}{2}}\ \boldsymbol{e}_x & s_{\frac{\theta}{2}}\ \boldsymbol{e}_y \\ -s_{\frac{\theta}{2}}\ \boldsymbol{e}_y & s_{\frac{\theta}{2}}\ \boldsymbol{e}_x & c_{\frac{\theta}{2}} & s_{\frac{\theta}{2}}\ \boldsymbol{e}_z \\ -s_{\frac{\theta}{2}}\ \boldsymbol{e}_x & -s_{\frac{\theta}{2}}\ \boldsymbol{e}_y & -s_{\frac{\theta}{2}}\ \boldsymbol{e}_z & c_{\frac{\theta}{2}} \end{pmatrix}, \tag{5.65}$$

$$\text{and} \quad \boldsymbol{Q}_r = \begin{pmatrix} c_{\frac{\theta}{2}} & s_{\frac{\theta}{2}}\ \boldsymbol{e}_z & -s_{\frac{\theta}{2}}\ \boldsymbol{e}_y & s_{\frac{\theta}{2}}\ \boldsymbol{e}_x \\ -s_{\frac{\theta}{2}}\ \boldsymbol{e}_z & c_{\frac{\theta}{2}} & s_{\frac{\theta}{2}}\ \boldsymbol{e}_x & s_{\frac{\theta}{2}}\ \boldsymbol{e}_y \\ s_{\frac{\theta}{2}}\ \boldsymbol{e}_y & -s_{\frac{\theta}{2}}\ \boldsymbol{e}_x & c_{\frac{\theta}{2}} & s_{\frac{\theta}{2}}\ \boldsymbol{e}_z \\ -s_{\frac{\theta}{2}}\ \boldsymbol{e}_x & -s_{\frac{\theta}{2}}\ \boldsymbol{e}_y & -s_{\frac{\theta}{2}}\ \boldsymbol{e}_z & c_{\frac{\theta}{2}} \end{pmatrix}. \tag{5.66}$$

Hence, Eq. (5.64) gives

$$\boldsymbol{p\prime} = \begin{pmatrix} c^2_{\frac{\theta}{2}} - s^2_{\frac{\theta}{2}}(\boldsymbol{e}_z^2 + \boldsymbol{e}_y^2 - \boldsymbol{e}_x^2) & -2(c_{\frac{\theta}{2}}s_{\frac{\theta}{2}}\boldsymbol{e}_z + s^2_{\frac{\theta}{2}}\boldsymbol{e}_x\boldsymbol{e}_y) & 2(c_{\frac{\theta}{2}}s_{\frac{\theta}{2}}\boldsymbol{e}_y + s^2_{\frac{\theta}{2}}\boldsymbol{e}_x\boldsymbol{e}_z) & 0 \\ 2(c_{\frac{\theta}{2}}s_{\frac{\theta}{2}}\boldsymbol{e}_z + s^2_{\frac{\theta}{2}}\boldsymbol{e}_x\boldsymbol{e}_y) & c^2_{\frac{\theta}{2}} - s^2_{\frac{\theta}{2}}(\boldsymbol{e}_z^2 + \boldsymbol{e}_x^2 - \boldsymbol{e}_y^2) & -2(c_{\frac{\theta}{2}}s_{\frac{\theta}{2}}\boldsymbol{e}_x - s^2_{\frac{\theta}{2}}\boldsymbol{e}_y\boldsymbol{e}_z) & 0 \\ -2(c_{\frac{\theta}{2}}s_{\frac{\theta}{2}}\boldsymbol{e}_y - s^2_{\frac{\theta}{2}}\boldsymbol{e}_x\boldsymbol{e}_z) & 2(c_{\frac{\theta}{2}}s_{\frac{\theta}{2}}\boldsymbol{e}_x + s^2_{\frac{\theta}{2}}\boldsymbol{e}_y\boldsymbol{e}_z) & c^2_{\frac{\theta}{2}} - s^2_{\frac{\theta}{2}}(\boldsymbol{e}_y^2 + \boldsymbol{e}_x^2 - \boldsymbol{e}_z^2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \boldsymbol{p}. \tag{5.67}$$

With the obvious substitutions $c^2_{\frac{\theta}{2}} - s^2_{\frac{\theta}{2}} = c_\theta$, and $2s^2_{\frac{\theta}{2}} = 1 - c_\theta = v_\theta$, this result is equivalent to Eq. (5.46), as could have been expected since this equation represents the rotation matrix for a rotation about the equivalent axis. Note that quaternion multiplication is *associative*, i.e., $\boldsymbol{q}^1(\boldsymbol{q}^2\boldsymbol{q}^3) = (\boldsymbol{q}^1\boldsymbol{q}^2)\boldsymbol{q}^3$. (Check this!)

5.5.4 Quaternion time rates and angular velocity

It was already mentioned earlier in this Chapter that the angular velocity three-vector cannot be calculated as the time derivative of any three-vector orientation representation. Also for quaternions the relationship between the time rate of the quaternion $\boldsymbol{q}$ and the corresponding angular velocity $\boldsymbol{\omega}$ is not trivial, and requires an *integrating factor*. However, the relation follows straightforwardly from Eqs (5.18) and (5.65):

$$\boxed{\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{\Omega}_l\ \boldsymbol{q},} \tag{5.68}$$

with $\boldsymbol{\Omega}$ the quaternion matrix corresponding to the quaternion vector $(\boldsymbol{\omega}^T\ 0)^T$.

Inverse relation The *inverse* of this relation follows from the following two observations:

1. $\boldsymbol{\Omega}_l\boldsymbol{q} = \boldsymbol{Q}_r(\boldsymbol{\omega}^T\ 0)^T$, Eq. (5.61), and
2. $\boldsymbol{Q}_r^{-1} = \boldsymbol{Q}_r^T$, Eq. (5.63).

Hence

$$\boxed{\begin{pmatrix}\boldsymbol{\omega}\\ 0\end{pmatrix} = 2\boldsymbol{Q}_r^T\dot{\boldsymbol{q}}.} \tag{5.69}$$

$2\boldsymbol{Q}_r^T$ is the integrating factor. Note that the forward and inverse relationships (5.68) and (5.69) *never* become singular, while this is not the case for the Euler angle sets, see, for example, Eq. (5.41).

References for this Chapter

[1] J. Angeles. *Rational Kinematics.* Springer Verlag, 1988.

[2] A. Bottema and B. Roth. *Theoretical Kinematics.* Dover Books on Engineering. Dover Publications, Inc., Mineola, NY, 1990.

[3] R. C. Buck and E. F. Buck. *Advanced calculus.* International series in pure and applied mathematics. McGraw-Hill, New York, NY, 3rd edition, 1978.

[4] H. Cheng and K. C. Gupta. An historical note on finite rotations. *Trans. ASME J. Appl. Mech.*, 56:139–145, 1989.

[5] H. C. Corben and P. Stehle. *Classical Mechanics.* Dover Publications, Inc., Mineola, NY, 2nd edition, 1994.

[6] M. J. Crowe. *A history of vector analysis: the evolution of the idea of a vectorial system.* University of Notre Dame Press, Notre Dame, IN, 1967.

[7] H. Goldstein. *Classical mechanics.* Addison-Wesley Series in Physics. Addison-Wesley, Reading, MA, 2nd edition, 1980.

[8] J. Gray. Olinde Rodrigues' paper of 1840 on transformation groups. *Archives History of Exact Sciences*, 21:375–385, 1980.

[9] H. W. Guggenheimer. *Differential Geometry.* Dover Publications, Inc., New York, NY, 1977.

[10] W. R. Hamilton. *Elements of quaternions*, volume I-II. Chelsea Publishing Company, New York, NY, 3rd edition, 1969. (Originally published in 1899).

[11] E. J. Haug. *Computer Aided Kinematics and Dynamics of Mechanical Systems. Volume 1: basic methods.* Series in Engineering. Allyn and Bacon, Boston, MA, 1989.

[12] S. Helgason. *Differential geometry, Lie groups, and symmetric spaces*, volume 80 of *Pure and Applied Mathematics.* Academic Press, New York, NY, 1978.

[13] B. P. Ickes. A new method for performing digital control system attitude computations using quaternions. *AIAA Journal*, 8(1):13–17, 1970.

[14] J. E. Marsden and T. S. Ratiu. *Introduction to Mechanics and Symmetry*, volume 17 of *Texts in Applied Mathematics*. Springer Verlag, New York, NY, 1994.

[15] J. M. McCarthy. *Introduction to Theoretical Kinematics*. MIT Press, Cambridge, MA, 1990.

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

[17] P. E. Nikravesh. *Computer-aided analysis of mechanical systems*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[18] P. E. Nikravesh, R. A. Wehage, and K. O. K. Euler parameters in computational kinematics and dynamics. part 1. *Trans. ASME J. Mech. Transm. Automation Design*, 107:358–365, 1985.

[19] F. C. Park. Distance metrics on the rigid-body motions with applications to mechanism design. *Trans. ASME J. Mech. Design*, 117:48–54, 1995.

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

[21] R. L. Pio. Euler angle transformations. *IEEE Trans. Autom. Control*, 11(4):707–715, 1966.

[22] P. D. Ritger and N. J. Rose. *Differential Equations with Applications*. McGraw-Hill, New York, NY, 1968.

[23] O. Rodrigues. Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace, et de la variation des co-ordonnées provenant de ses déplacements considérés indépendamment des causes qui peuvent les produire. *Journal de Mathématiques Pures et Appliquées*, 5:380–440, 1840.

[24] W. Rudin. *Principles of mathematical analysis*. International series in pure and applied mathematics. McGraw-Hill Kogakusha, Tokyo, Japan, 3rd edition, 1976.

[25] A. E. Samuel, P. R. McAree, and K. H. Hunt. Unifying screw geometry and matrix transformations. *Int. J. Robotics Research*, 10(5):454–472, 1991.

[26] K. W. Spring. Euler parameters and the use of quaternion algebra in the manipulation of finite rotations: a review. *Mechanism and Machine Theory*, 21(5):365–373, 1986.

[27] G. Strang. *Calculus*. Wellesley-Cambridge Press, Wellesley, MA, 1991.

[28] J. Stuelpnagel. On the parametrization of the three-dimensional rotation group. *SIAM Review*, 6(4):422–430, 1964.

[29] C. A. Truesdell. Influence of elasticity on analysis. *Bulletin of the AMS*, 9(3):293–310, 1983.

Chapter 6

Pose coordinates

6.1 Introduction

A rigid body in E^3 has six degrees of freedom: three in translation and three in rotation. Chapter 5 has discussed these last three degrees of freedom separately; this Chapter integrates them with the translations, following much the same route as in Chapter 5.

Coordinate representations containing six parameters have been developed over the years. However, the fundamental geometric properties of rigid body motion as described in Chapter 3 cannot be represented by classical six-vector vector spaces (i.e., with addition and/or multiplication of six-vectors as the standard operators in these spaces). One noteworthy exception is the *velocity* of a rigid body: this *can* be represented by a six-vector, i.e., a screw, or rather, a twist. Recall from Chapter 3 that the definition of velocity and acceleration used in this text is the following: the minimum information one needs to find the velocity and acceleration of any point moving together with the body. So, this Chapter describes what this minimum information is for different coordinate motion representations.

One often uses a *non-minimal matrix* representation to represent the properties of rigid body motion. The same trade-offs exist as in the previous Chapters: improved properties on the one hand, but extra cost because of the need to carry along a number of constraints on the other hand.

6.2 Homogeneous transform

Orientations and their representations are not very intuitive in more than one respect. However, extending the description to include translations turns out to require only a minor extra effort.

6.2.1 Definition and use

Fact-to-Remember 38 (Pose representation)
The pose (i.e., relative position and orientation) of a frame $\{b\}$ with respect to a frame $\{a\}$ can be represented by (i) the position vector ${}_a\boldsymbol{p}^{a,b}$ of the origin of $\{b\}$ with respect to the origin of $\{a\}$ and expressed with respect to $\{a\}$, plus (ii) the orientation matrix ${}^b_a\boldsymbol{R}$ of $\{b\}$ with respect to $\{a\}$. These are often combined into a 4×4 pose (matrix) ${}^b_a\boldsymbol{T}$ (or homogeneous transformation matrix, or homogeneous transform, for short):

$$\boxed{{}^b_a\boldsymbol{T} \triangleq \begin{pmatrix} {}^b_a\boldsymbol{R} & {}_a\boldsymbol{p}^{a,b} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix}.} \qquad (6.1)$$

${}^b_a\boldsymbol{T}$ is the coordinate representation of a *point* in SE(3), or, equivalently, the representation of a *frame* in E^3. Although at first sight it might look a bit strange, this matrix representation is particularly interesting since, if the coordinates of a point $\boldsymbol{p}$ are known with respect to $\{b\}$ (i.e., the coordinate vector ${}_b\boldsymbol{p}$ is known), the point's coordinates with respect to $\{a\}$ (i.e., ${}_a\boldsymbol{p}$) are calculated as

$$\begin{pmatrix} {}_a\boldsymbol{p} \\ 1 \end{pmatrix} = {}^b_a\boldsymbol{T} \begin{pmatrix} {}_b\boldsymbol{p} \\ 1 \end{pmatrix}. \qquad (6.2)$$

This is obvious from Fig. 6.1. Note that Eq. (6.2) extends the position three-vectors ${}_a\boldsymbol{p}$ and ${}_b\boldsymbol{p}$ into *four-vectors*, by adding a constant "1" row, i.e., the vectors are made *homogeneous*. Hence, the name of this pose representation.

6.2.2 Active and passive interpretation

As for rotations and orientations (Sect. 5.2.3), one can interpret a homogeneous transformation matrix both actively and passively. The passive interpretation is often connected to the terminology "pose," while the termi-

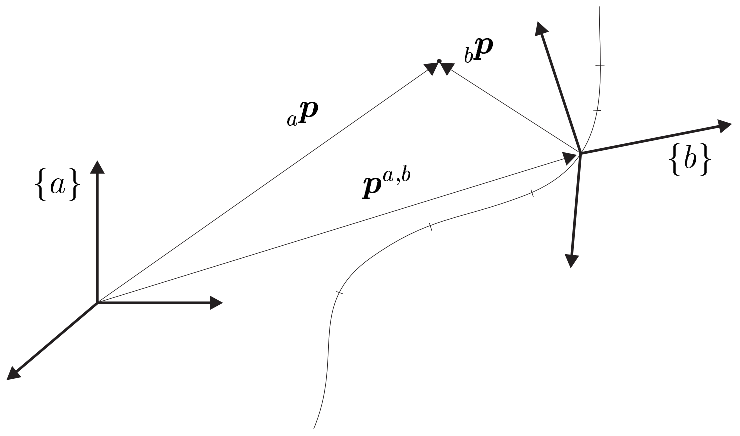


Figure 6.1: Frame $\{b\}$ moves with respect to frame $\{a\}$. The point $\boldsymbol{p}$ moves together with$\{b\}$.

nology "displacement" suggests an active interpretation.

6.2.3 Uniqueness

From the definition (6.1) of the homogeneous transformation matrix, and the uniqueness property of the rotation matrix (Fact. 25), the following fact is obvious:

Fact-to-Remember 39 (Uniqueness)
A homogeneous transformation matrix is a unique and unambiguous representation of the relative pose of two right-handed, orthogonal reference frames in the Euclidean space E^3. This means that one single homogeneous transformation matrix corresponds to each relative pose, and each homogeneous transformation matrix represents one single relative pose.

6.2.4 Inverse

Given the simple formula for the inverse of a rotation matrix, Eq. (5.7), it is straightforward to check that the *inverse* of a homogeneous transformation matrix ${}^{b}_{a}\boldsymbol{T}$ is

$$\boxed{{}^{b}_{a}\boldsymbol{T}^{-1} = \begin{pmatrix} {}^{b}_{a}\boldsymbol{R}^{T} & -{}^{b}_{a}\boldsymbol{R}^{T}\,{}_{a}\boldsymbol{p}^{a,b} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix}.} \tag{6.3}$$

Note that constructing ${}^{b}_{a}\boldsymbol{T}^{-1}$ from ${}^{b}_{a}\boldsymbol{T}$ needs nothing more complicated than one simple matrix multiplication.

6.2.5 Non-minimal representation

A direct consequence of the non-minimality of the rotation matrix (Fact 27) is that

Fact-to-Remember 40 (Non-minimal representation)
A homogeneous transformation matrix is not a minimal representation of a pose. Again, the advantage is that it has <u>no coordinate singularities</u>.

6.2.6 Isometry—SE(3)

Just as rotation matrices (Sect. 5.2.7), homogeneous transformation matrices are *isometries* of the Euclidean space, since they maintain angles between vectors, and lengths of vectors. Moreover, right-handed frames are mapped into right-handed frames, and the determinant of homogeneous transformation matrices is +1. Their algebraic properties correspond to those of the Lie group SE(3), Sect. 3.3.

6.2.7 Compound poses

The same reasoning as in Sect. 5.2.8 leads straightforwardly to the formula for composition of pose transforms. For example, knowing the pose ${}^{tl}_{ee}\boldsymbol{T}$ of the "tool" frame $\{tl\}$ with respect to the "end effector" frame $\{ee\}$, and

${}^{ee}_{bs}\boldsymbol{T}$ of frame $\{ee\}$ with respect to the “base” frame $\{bs\}$, gives the pose ${}^{tl}_{bs}\boldsymbol{T}$ of the tool with respect to the base as

$$\boxed{{}^{tl}_{bs}\boldsymbol{T} = {}^{ee}_{bs}\boldsymbol{T}\,{}^{tl}_{ee}\boldsymbol{T}.} \tag{6.4}$$

This equation is easily checked, for example, by calculating the coordinates of a point in three reference frames, i.e., a procedure similar to the one used to derive Eq. (6.2) and illustrated in Fig. 6.1.

6.3 Finite displacement twist

Equation (6.1) uses a 4×4 matrix to represent a pose. Of course, other orientation representations could be used instead of the rotation matrix. Hence, one frequently encountered alternative for a homogeneous transformation matrix is the *finite displacement twist*, which is the following six-vector:

$$\mathbf{t}_d = \begin{pmatrix}\alpha \; \beta \; \gamma \; x \; y \; z\end{pmatrix}^T, \tag{6.5}$$

with α, β, and γ a set of Euler angles (of any possible type), and x, y, and z the coordinates of a reference point on the rigid body. Recall however, from Section 3.3, that the finite displacement twist is *not* a screw: the first three components are an Euler angle set, which is not a member of a vector space, Sect. 5.3.4. On the other hand, the *infinitesimal* displacement twist is a screw, Eq. (4.15).

6.4 Time derivative of pose—Derivative of homogeneous transform

Figure 6.1 sketches a moving rigid body, or rather the motion of the reference frame $\{b\}$ attached to this body. The origin of this frame traces a curve in E^3; equivalently, the rigid body traces a curve in SE(3). This Section is interested in representing the first order kinematics (i.e., the velocity) of the moving body. To this end, consider an arbitrary point $\boldsymbol{p}$ that moves together with the moving body. The coordinates of this point with respect to $\{b\}$ are known and constant; its coordinates with respect to the world frame $\{a\}$ are found from Eq. (6.2):

$${}_a\boldsymbol{p} = {}^b_a\boldsymbol{R}\,{}_b\boldsymbol{p} + {}_a\boldsymbol{p}^{a,b}.$$

The time derivative of this coordinate transformation (i.e., of the left-translated curve) is straightforward to calculate, given the time derivative of the rotation matrix, Eq. (5.20):

$$\begin{aligned}
{}_a\dot{\boldsymbol{p}} &= {}^b_a\dot{\boldsymbol{R}}\,{}_b\boldsymbol{p} + {}_a\dot{\boldsymbol{p}}^{a,b}, & (6.6)\\
&= {}^b_a\dot{\boldsymbol{R}}\,\left({}^b_a\boldsymbol{R}^T({}_a\boldsymbol{p} - {}_a\boldsymbol{p}^{a,b})\right) + {}_a\dot{\boldsymbol{p}}^{a,b}, &\\
&= [{}_a\boldsymbol{\omega}]\,{}_a\boldsymbol{p} + {}_a\dot{\boldsymbol{p}}^{a,b} - [{}_a\boldsymbol{\omega}]\,{}_a\boldsymbol{p}^{a,b}. & (6.7)
\end{aligned}$$

$\boldsymbol{\omega}$ is the angular velocity three-vector of the moving body. $[\boldsymbol{\omega}]$ is the skew-symmetric matrix corresponding to taking the vector product with $\boldsymbol{\omega}$, Eq. (5.19). Hence:

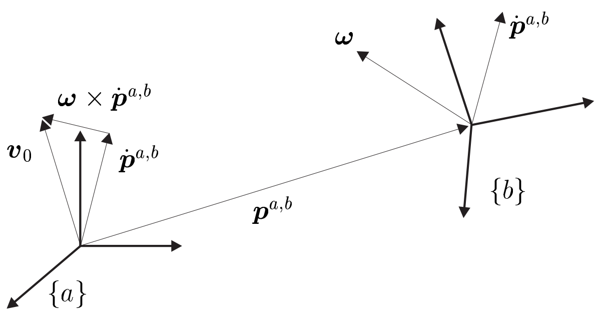
$$\begin{pmatrix}{}_a\dot{\boldsymbol{p}}\\ 0\end{pmatrix} = {}^b_a\dot{\boldsymbol{T}}\,{}^b_a\boldsymbol{T}^{-1}\begin{pmatrix}{}_a\boldsymbol{p}\\ 1\end{pmatrix}, \quad \text{with} \quad {}^b_a\dot{\boldsymbol{T}} = \begin{pmatrix}{}^b_a\dot{\boldsymbol{R}} & {}_a\dot{\boldsymbol{p}}^{a,b}\\ \boldsymbol{0}_{1\times 3} & 0\end{pmatrix}. \tag{6.8}$$

(Compare to Eq. (5.17).) In Eq. (6.8), the operator $\dot{\boldsymbol{T}}\,\boldsymbol{T}^{-1}$ works linearly on the coordinates of the point fixed to the moving body. Equation (6.7) can also be written as

$${}_a\dot{\boldsymbol{p}} = [{}_a\boldsymbol{\omega}]\,{}_a\boldsymbol{p} + {}_a\boldsymbol{v}_0, \tag{6.9}$$

with ${}_a\boldsymbol{v}_0 \triangleq {}_a\dot{\boldsymbol{p}}^{a,b} - [{}_a\boldsymbol{\omega}]\,{}_a\boldsymbol{p}^{a,b}$ the velocity of the point on the moving body that *instantaneously coincides* with the *origin* of the reference frame $\{a\}$ (Fig. 6.2). This yields a relationship that is similar to the corresponding relationship Eq. (5.20) for rotations:

$$\boxed{{}^b_a\dot{\boldsymbol{T}}\,{}^b_a\boldsymbol{T}^{-1} = \begin{pmatrix} [{}_a\boldsymbol{\omega}] & {}_a\boldsymbol{v}_0 \\ \boldsymbol{0}_{1\times 3} & 0 \end{pmatrix}.} \tag{6.10}$$



$\boldsymbol{\omega}$
$\dot{\boldsymbol{p}}^{a,b}$
$\boldsymbol{\omega} \times \dot{\boldsymbol{p}}^{a,b}$
$\boldsymbol{v}_0$
$\dot{\boldsymbol{p}}^{a,b}$
$\{b\}$
$\boldsymbol{p}^{a,b}$
$\{a\}$

Figure 6.2: $\boldsymbol{v}_0$ is the velocity of the point of the moving rigid body that instantaneously coincides with the origin of the world frame $\{a\}$. It is the sum of (i) the translational velocity of the origin of the moving frame $\{b\}$, and (ii) the translational velocity generated by the angular velocity $\boldsymbol{\omega}$ working on the moving body at a distance $\boldsymbol{p}^{a,b}$ from the origin.

The determinant of $[\boldsymbol{\omega}]$ is always zero. This means that, for a general motion, there is no point $\boldsymbol{p}$ with zero velocity $\dot{\boldsymbol{p}}$. Indeed, the set of three linear equations in Eq. (6.9) doesn't have a solution $\boldsymbol{p}$ for $\dot{\boldsymbol{p}} = 0$, since the coefficient matrix of $\boldsymbol{p}$ is not of full rank.

The factor $\boldsymbol{T}^{-1}$ corresponds to the *left translation* of the tangent vector $\dot{\boldsymbol{T}}$ to the origin, Sect. 3.4 (although it appears on the right-hand side of the product). One can follow a similar reasoning, but now expressing the velocity of the moving point with respect to the body-fixed reference frame $\{b\}$. This would yield an element of *se(3)* by *right translation* of $\dot{\boldsymbol{T}}$: $\boldsymbol{T}^{-1}\dot{\boldsymbol{T}}$.

6.5 Time derivative of pose—Twists

Although $\dot{\boldsymbol{T}}\boldsymbol{T}^{-1}$ in Eq. (6.10) is a 4×4 matrix, its complete information contents can be represented in *two* three-vectors: $\boldsymbol{\omega}$ and $\boldsymbol{v}_0$. These two three-vectors together form the six-vector $\mathbf{t} = (\boldsymbol{\omega}^T\ \boldsymbol{v}_0^T)^T$ that was called a (screw) *twist* in Sect. 3.8, and that is a member of se(3), the tangent space to SE(3) at the identity element. We represent a screw twist by the following six-vector:

$$\mathbf{t} = \begin{pmatrix} \boldsymbol{\omega} \\ \boldsymbol{v}_0 \end{pmatrix}. \tag{6.11}$$

In this representation, adding rigid body velocities corresponds to adding twist vectors. A second six-vector alternative for representing rigid body velocity is extracted from $\dot{\boldsymbol{T}}$ in Eq. (6.8):

$$\mathbf{t} = \begin{pmatrix} \boldsymbol{\omega} \\ \dot{\boldsymbol{p}}^{a,b} \end{pmatrix}. \tag{6.12}$$

The fact that this six-vector comes directly from the time derivative of a homogeneous transformation matrix (or pose), inspired the following (non-standard) terminology:

Fact-to-Remember 41 (Screw twist vs. Pose twist)
The linear velocity three-vector in a pose twist represents the translational velocity of the origin of $\{b\}$ *with respect to the origin of the world reference frame* $\{a\}$*, Eq. (6.12).*
The linear velocity three-vector in a screw twist represents the velocity of the point that instantaneously coincides with the origin of $\{a\}$*, Eq. (6.11).*
A body-fixed twist is a screw twist for which the world reference frame instantaneously coincides with $\{b\}$*.*

Pose twists and body-fixed twists are probably more often used in robot control software than screw twists. A closer look at all these twists reveals why "screw twists" (i.e., members of the "tangent space at the identity," Sect. 3.4) are more appropriate (from a computational view) than "pose twists" (i.e., members of the tangent spaces at arbitrary elements on SE(3)):

Fact-to-Remember 42 (Screw twist vs. Pose twist (cont'd))
From the six numbers in the screw twist, one can deduce the position and orientation of the instantaneous screw axis; this is not possible when using the six numbers in the pose twist.

Indeed, take the *scalar* product of both three-vectors in the twist. For a screw twist, this gives:

$$\boldsymbol{\omega} \cdot \left(\dot{\boldsymbol{p}}^{a,b} + \boldsymbol{p}^{a,b} \times \boldsymbol{\omega}\right) = \boldsymbol{\omega} \cdot \dot{\boldsymbol{p}}^{a,b},$$

since the vector product is always orthogonal to $\boldsymbol{\omega}$. If one takes the position vector $\boldsymbol{p}^{a,b}$ as the vector of the point on the ISA *closest to the origin*, then $\boldsymbol{\omega}$ and $\dot{\boldsymbol{p}}^{a,b}$ are parallel: the angular velocity is parallel to the linear velocity of a point on the ISA, since both velocities are parallel to the ISA. Hence, the scalar product above gives the projection of $\dot{\boldsymbol{p}}^{a,b}$ on $\boldsymbol{\omega}$, and since both are parallel this yields all information about $\dot{\boldsymbol{p}}^{a,b}$ too. So, the complete motion information is found in the six numbers of the screw twist: the position and orientation of the ISA, and the angular and linear velocities on this ISA.

A similar scalar product operation on the pose twist cannot give information about the ISA, since one can say nothing about the relative orientation of $\boldsymbol{\omega}$ and $\dot{\boldsymbol{p}}^{a,b}$.

Taking the *vector* product (instead of the scalar product, as done above) of both three-vectors in the twist gives:

$$\boldsymbol{\omega} \times \left(\dot{\boldsymbol{p}}^{a,b} + \boldsymbol{p}^{a,b} \times \boldsymbol{\omega}\right) = -[\boldsymbol{\omega}][\boldsymbol{\omega}]\boldsymbol{p}^{a,b}.$$

The left-hand side is a known tree-vector. So, this might suggest that one can solve for $\boldsymbol{p}^{a,b}$, but this is not the case: the matrix $[\boldsymbol{\omega}]$ has vanishing determinant, and is hence not of full rank and not invertible.

In summary, the pose twist can only be used to construct the ISA if the three-vector $\boldsymbol{p}^{a,b}$ is given as *extra* information. The terminology "screw twist," "pose twist," and "body-fixed twist" is not in general use: most references just use one of these three representations and call it a "twist" (or "(generalized) velocity"), without explicitly telling the reader which one is being used.

6.5.1 Order of three-vectors in twist representation

When reading the robotics literature, it is important to know that different authors use a different *order* of the three-vectors in the coordinate representations of twists and wrenches:

$$\mathbf{t} = \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \end{pmatrix}, \quad \text{and} \quad \mathbf{w} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{m} \end{pmatrix}, \tag{6.13}$$

or

$$\mathbf{t} = \begin{pmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{pmatrix}, \quad \text{and} \quad \mathbf{w} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{m} \end{pmatrix}. \tag{6.14}$$

The order is not a *structural* property of rigid body twists! This text uses the last convention. The reason is that with this representation, screw twists and screw wrenches transform in exactly the same way, i.e., as screws; in the alternative representation (6.13), one must introduce two different transformation matrices for twists and wrenches. This has, however, also an important advantage: twists and wrenches *are* two different things (Sect. 3.9), and using different transformation equations for both emphasizes this difference.

6.5.2 Invariants

The previous paragraphs explain some differences between *coordinate representations*, but one should not lose sight of the fact that they all describe the same *physical* motion. Hence, they all have the same geometrical *invariants*, as introduced already in Chapter 3:

Invariants of rigid body motion. Physically, the finite or instantaneous motion of a rigid body is represented by the following *invariants*: (i) the screw axis, (ii) the translation or translational velocity vector of a point *on the screw axis*, (ii) the angular rotation or rotation velocity vector about the screw axis, and (iv) the ratio of the translational and angular vectors, which is called the *pitch.* Recall that the pitch has the physical dimensions of length. "Invariant" means that these things don't change under (i) a change of reference frame, (ii) a change of coordinate twist representation (e.g., screw twist to pose twist), and (iii) a change of physical units (e.g., meters to inches).

Invariants of force on rigid body. Physically, the force and moment exerted on a rigid body is represented by the following *invariants*: (1) the screw axis, (2) the linear force vector acting on the body, (3) the torque felt in a point on the screw axis, and (4) the pitch which is the ratio of the torque and force vectors.

6.5.3 Exponential and logarithm

Section 3.5 introduced the concept of the *exponentiation* that maps a twist (i.e., a rigid body velocity) onto a finite displacement (i.e., a pose): $\exp : \text{se}(3) \rightarrow \text{SE}(3), \mathbf{t} \mapsto \boldsymbol{T}$. Equation (5.21) gave a coordinate representation for the exponential map in the case that the twist is a pure rotation. Since the time derivative of screw twists (or rather, of the corresponding 4×4 matrix) obey the same differential equation, Eq. (6.10), as the time derivative of rotation matrices, Eq. (5.20), a similar exponentiation formula works for twists and displacements too: the matrix exponential of the matrix corresponding to a *screw* twist $\mathbf{t} = ({}_a\boldsymbol{\omega}^T \; {}_a\boldsymbol{v}_0^T)^T \triangleq (\omega_x \, \omega_y \, \omega_z \, v_x \, v_y \, v_z)^T$ is the pose $\boldsymbol{T}$:

$$\boxed{\boldsymbol{T} = \exp \begin{pmatrix} [{}_a\boldsymbol{\omega}] & {}_a\boldsymbol{v}_0 \\ \mathbf{0}_{1\times 3} & 0 \end{pmatrix}.} \tag{6.15}$$

This works for *screw* twists only, since the exponential is only well defined on se(3).

The *logarithm of a finite displacement* is also a well-defined, hence *structural*, operation, [14, p. 414]. The result of the logarithm operation on a finite displacement is the screw twist that generates this displacement in one unit of time. When using a homogeneous transformation matrix for the displacement, the logarithm of this matrix gives the screw twist in the form of the argument of the exponential function in Eq. (6.15).

6.5.4 Canonical coordinates

The previous Section showed how to represent a finite displacement as the exponential of a twist. This approach leads to two different sets of so-called *canonical coordinates*:

1. The six *canonical coordinates of the first kind*, [9, 14, 18], represent the velocity that must be given to the world reference frame in order to make it coincide, after one unit of time, with the reference frame on the rigid body at its current pose.

2. The six *canonical coordinates of the second kind* represent the same displacement as the composition of six elementary exponentiations:

$$\begin{aligned} \boldsymbol{T} = &\exp\left((\omega'_x\ 0\ 0\ 0\ 0\ 0)^T\right)\\ &\exp\left((0\ \omega'_y\ 0\ 0\ 0\ 0)^T\right)\\ &\exp\left((0\ 0\ \omega'_z\ 0\ 0\ 0)^T\right)\\ &\exp\left((0\ 0\ 0\ v'_x\ 0\ 0)^T\right)\\ &\exp\left((0\ 0\ 0\ 0\ v'_y\ 0)^T\right)\\ &\exp\left((0\ 0\ 0\ 0\ 0\ v'_z)^T\right). \end{aligned} \tag{6.16}$$

   This is an example of the composition of transformation matrices, Eq. (6.4), since each of the exponentiations gives a transformation matrix. The first three give pure rotations (about the moving axes of an orthogonal frame, hence an XYZ Euler angle representation of the orientation), and the last three are pure translations (also along moved axes):

$$\boldsymbol{T} = \boldsymbol{R}(X,\omega'_x)\boldsymbol{R}(Y,\omega'_y)\boldsymbol{R}(Z,\omega'_z)\boldsymbol{Tr}(X,v'_x)\boldsymbol{Tr}(Y,v'_y)\boldsymbol{Tr}(Z,v'_z), \tag{6.17}$$

   with $\boldsymbol{R}(X,\omega'_x)$ the homogeneous transformation matrix corresponding to a pure rotation about the X axis over an angle ω'_x, and $\boldsymbol{Tr}(X,v'_x)$ the homogeneous transformation matrix corresponding to a pure translation along the X axis over a distance v'_x. (This distance is the product of velocity with time, but the time period is "1," by definition.)

6.5.5 Infinitesimal displacement

As in most other engineering sciences, robotics often uses "infinitesimal" quantities to describe geometric entities that are "very close" to each other. The *infinitesimal displacement twist*, denoted by $\mathbf{t}_\Delta$, and the *infinitesimal transformation matrix*, denoted by $\boldsymbol{T}_\Delta$, both describe small differences in *pose*. As in the case of small rotations (Sect. 5.2.11), these infinitesimal displacements are derived by stopping the Taylor series of the exponential in Eq. (6.15) after the linear term. This gives:

$$\boxed{\mathbf{t}_\Delta = \begin{pmatrix}\delta_x\\ \delta_y\\ \delta_z\\ d_x\\ d_y\\ d_z\end{pmatrix}, \qquad \boldsymbol{T}_\Delta = \begin{pmatrix}1 & -\delta_z & \delta_y & d_x\\ \delta_z & 1 & -\delta_x & d_y\\ -\delta_y & \delta_x & 1 & d_z\\ 0 & 0 & 0 & 1\end{pmatrix}.} \tag{6.18}$$

d_x, d_y and d_z are small translations along the X, Y and Z axes, respectively; δ_x, δ_y and δ_z are small rotations about the axes. The exponential used above is only well defined for screw twists, Eq. (6.11). However, infinitesimal pose twists can be *defined* by the same equations (6.18); the meaning of the three-vector $(d_x\, d_y\, d_z)^T$ changes accordingly.

6.6 Representation transformation

This Section describes how the coordinate representations of the geometric entities introduced in this Chapter and in previous Chapters transform under a change of reference frame. These transformations are important in the kinematic and dynamic descriptions of robotic devices. For example, calculating the end effector velocity of a serial robot arm when all joint velocities are given, requires the summation of the end effector velocities caused by each joint independently: these independent joint velocities are easily expressed in the reference frames at the joints themselves, but must then be transformed to a common world reference frame before they can be added.

Another important motivation to study the transformations between different coordinate representations of the same structural concepts is the principle of *invariance*: theoretical and/or practical results derived in the framework of one particular coordinate representation should describe the same things when transformed into another coordinate representation. This statement might seem trivial, but nevertheless many publications in the robotics literature violate this principle, [5, 6, 13]. Minimum sets of invariants were summarized in Sect. 6.5.2.

For rigid body entities, two reference changes are relevant: (i) a change of the world reference frame, or (ii) a change of the rigid body reference frame, Fig. 6.3.

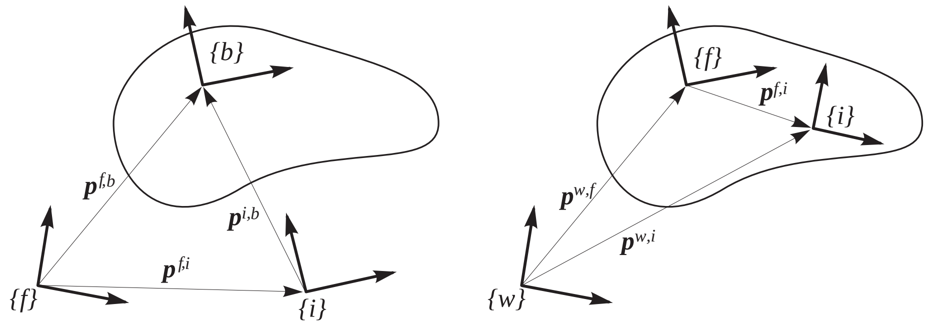


Figure 6.3: Transformations of screw-like geometric entities under a change of world and body-fixed reference frames, respectively.

6.6.1 Three-vector transformation

Free three-vectors have the simplest transformation: if the "initial" world reference frame $\{i\}$ is described with respect to a "final" world reference frame $\{f\}$ through the homogeneous transformation matrix

$$^{i}_{f}\boldsymbol{T} = \begin{pmatrix} ^{i}_{f}\boldsymbol{R} & _{f}\boldsymbol{p}^{f,i} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix},$$

the *components* of the free three-vector $\boldsymbol{v}$ transform as

$$_{f}\boldsymbol{v} = {}^{i}_{f}\boldsymbol{R}\ {}_{i}\boldsymbol{v}. \tag{6.19}$$

Point vector components transform as

$$\begin{pmatrix} _{f}\boldsymbol{p} \\ 1 \end{pmatrix} = {}^{i}_{f}\boldsymbol{T} \begin{pmatrix} _{i}\boldsymbol{p} \\ 1 \end{pmatrix}. \tag{6.20}$$

Note that the *physical* vectors don't change, but only their *coordinates*.

6.6.2 Line transformation

A line in Plücker coordinates has a representation $\mathcal{L}_{\text{pl}}(\boldsymbol{d},\boldsymbol{m})$. Denote the vector from the origin of the frame $\{i\}$ to the closest point on the line by $\boldsymbol{p}^{i,l}$, and similarly denote the vector from the origin of the frame $\{f\}$ to the closest point on the line by $\boldsymbol{p}^{f,l}$. Then, $\boldsymbol{p}^{i,l} = \boldsymbol{d} \times \boldsymbol{m}/(\boldsymbol{d}\cdot\boldsymbol{d})$, Eq. (4.6). Changing the world reference frame from $\{i\}$ to $\{f\}$ implies the following transformations:

1. The direction vector $\boldsymbol{d}$ does not change *physically*, but its components change if the frames $\{i\}$ and $\{f\}$ are not *parallel*:
$$ {}_f\boldsymbol{d} = {}^i_f\boldsymbol{R}\; {}_i\boldsymbol{d}. \tag{6.21}$$
2. The moment vector $\boldsymbol{m}$ changes (the physical three-vector, as well as its coordinates) if the frames have a *different origin*:
$$\begin{aligned} {}_f\boldsymbol{m} &= {}_f\boldsymbol{p}^{f,l} \times {}_f\boldsymbol{d} \\ &= \left({}_f\boldsymbol{p}^{f,i} + {}_f\boldsymbol{p}^{i,l}\right) \times {}_f\boldsymbol{d} \\ &= {}_f\boldsymbol{p}^{f,i} \times \left({}^i_f\boldsymbol{R}\; {}_i\boldsymbol{d}\right) + {}^i_f\boldsymbol{R}\; \left({}_i\boldsymbol{p}^{i,l} \times {}_i\boldsymbol{d}\right) \\ &= \left[{}_f\boldsymbol{p}^{f,i}\right] {}^i_f\boldsymbol{R}\; {}_i\boldsymbol{d} + {}^i_f\boldsymbol{R}\,{}_i\boldsymbol{m}. \end{aligned} \tag{6.22}$$

Combining the transformations (6.21) and (6.22) gives

$$\boxed{\begin{pmatrix} {}_f\boldsymbol{d} \\ {}_f\boldsymbol{m} \end{pmatrix} = \begin{pmatrix} {}^i_f\boldsymbol{R} & \boldsymbol{0}_3 \\ \left[{}_f\boldsymbol{p}^{f,i}\right] {}^i_f\boldsymbol{R} & {}^i_f\boldsymbol{R} \end{pmatrix} \begin{pmatrix} {}_i\boldsymbol{d} \\ {}_i\boldsymbol{m} \end{pmatrix}.} \tag{6.23}$$

Recall that $[\boldsymbol{p}]$ denotes the skew-symmetric matrix that corresponds to taking the vector product with the three-vector $\boldsymbol{p}$, Eq. (5.19).

6.6.3 Screw twist transformation

A screw $\mathcal{L}_{\text{sc}}(\boldsymbol{d},\boldsymbol{v})$ consists of two three-vectors bound to a line. Its coordinates with respect to a reference frame $\{i\}$ have been represented as, Eq. (4.12),

$$ {}_i\mathsf{s} = \begin{pmatrix} {}_i\boldsymbol{d} \\ {}_i\boldsymbol{p}^{i,l} \times {}_i\boldsymbol{d} + {}_i\boldsymbol{v} \end{pmatrix},$$

with $\boldsymbol{p}^{i,l}$ the vector from the origin of reference frame $\{i\}$ to a point on the screw axis.

Change of world reference frame. As for the line, the components of the direction vector $\boldsymbol{d}$ change according to Eq. (6.21), under a change of world reference frame from $\{i\}$ to $\{f\}$: ${}_f\boldsymbol{d} = {}^i_f\boldsymbol{R}\; {}_i\boldsymbol{d}$. The moment components of the screw coordinates transform as

$$\begin{aligned} {}_f\boldsymbol{m} &= {}_f\boldsymbol{p}^{f,l} \times {}_f\boldsymbol{d} + {}_f\boldsymbol{v} \\ &= \left({}_f\boldsymbol{p}^{f,i} + {}_f\boldsymbol{p}^{i,l}\right) \times {}_f\boldsymbol{d} + {}_f\boldsymbol{v} \\ &= {}_f\boldsymbol{p}^{f,i} \times \left({}^i_f\boldsymbol{R}\; {}_i\boldsymbol{d}\right) + {}^i_f\boldsymbol{R}\left({}_i\boldsymbol{p}^{i,l} \times {}_i\boldsymbol{d} + {}_i\boldsymbol{v}\right). \end{aligned} \tag{6.24}$$

Hence, the transformation matrix of the screw coordinate six-vector turns out to be exactly the same as for the transformation of a line, [11, 15, 16, 25, 26]. This text calls it the (finite) *screw transformation matrix* ${}^{i}_{f}\boldsymbol{S}$ (or *screw transform* for short):

$$\boxed{{}^{i}_{f}\boldsymbol{S} = \begin{pmatrix} {}^{i}_{f}\boldsymbol{R} & \boldsymbol{0}_3 \\ \left[{}_{f}\boldsymbol{p}^{f,i}\right] {}^{i}_{f}\boldsymbol{R} & {}^{i}_{f}\boldsymbol{R} \end{pmatrix}.} \tag{6.25}$$

(This name is not standardized!) Since (screw) twists, infinitesimal displacement twists, and wrenches are all instantiations of a screw, their coordinates all transform with the same screw transformation matrix ${}^{i}_{f}\boldsymbol{S}$ from frame $\{i\}$ to $\{f\}$. Note that:

1. The screw transform can be built from the homogeneous transform with only one matrix multiplication.
2. A *finite* displacement twist is *not* a screw, and hence does not transform in this manner.
3. Every screw transformation matrix has unit determinant.
4. The fact that this text chooses the same screw representation for twists and wrenches is not a structural property, but the consequence of an *arbitrary* choice.

Inverse of screw transformation matrix. By definition, the *inverse* of ${}^{i}_{f}\boldsymbol{S}$ is given by

$$\left({}^{i}_{f}\boldsymbol{S}\right)^{-1} = {}^{f}_{i}\boldsymbol{S} = \begin{pmatrix} {}^{f}_{i}\boldsymbol{R} & \boldsymbol{0}_3 \\ \left[{}_{i}\boldsymbol{p}^{i,f}\right] {}^{f}_{i}\boldsymbol{R} & {}^{f}_{i}\boldsymbol{R} \end{pmatrix}. \tag{6.26}$$

It is easy to check that this is equal to

$$\boxed{\left({}^{i}_{f}\boldsymbol{S}\right)^{-1} = \widetilde{\boldsymbol{\Delta}}\, {}^{i}_{f}\boldsymbol{S}^{T}\, \widetilde{\boldsymbol{\Delta}},} \tag{6.27}$$

with $\widetilde{\boldsymbol{\Delta}}$ as in Eq. (4.18). Equation (6.27) is sometimes called the *spatial transpose*, [7, 11], and denoted by ${}^{i}_{f}\boldsymbol{S}'$ or ${}^{i}_{f}\boldsymbol{S}^{S}$:

$$\text{if} \quad \boldsymbol{S} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix} \quad \text{then} \quad \boldsymbol{S}^{S} = \begin{pmatrix} \boldsymbol{D}^{T} & \boldsymbol{B}^{T} \\ \boldsymbol{C}^{T} & \boldsymbol{A}^{T} \end{pmatrix}. \tag{6.28}$$

This definition is attractive in the sence that the 6×6 screw transformation matrix is then *spatially orthogonal*: it has the same orthogonality conditions as the 3×3 rotation matrix (Sect. 5.2, Fact 26).

Infinitesimal transformation If the change in the world reference frame is only infinitesimal (i.e., the frame changes over an infinitesimal displacement $\mathbf{t}_{\Delta} = (\delta_x\, \delta_y\, \delta_z\, d_x\, d_y\, d_z)^{T}$) the screw transformation matrix $\boldsymbol{S}$ in Eq. (6.25) becomes an *infinitesimal screw transformation matrix* $\boldsymbol{S}_{\Delta}$, [24, 26]:

$$\boldsymbol{S}_{\Delta}(\mathbf{t}_{\Delta}) = \begin{pmatrix} 1 & -\delta_z & \delta_y & 0 & 0 & 0 \\ \delta_z & 1 & -\delta_x & 0 & 0 & 0 \\ -\delta_y & \delta_x & 1 & 0 & 0 & 0 \\ 0 & -d_z & d_y & 1 & -\delta_z & \delta_y \\ d_z & 0 & -d_x & \delta_z & 1 & -\delta_x \\ -d_y & d_x & 0 & -\delta_y & \delta_x & 1 \end{pmatrix}. \tag{6.29}$$

Active and passive interpretations $\boldsymbol{S}$ and $\boldsymbol{S}_\Delta$ work on screws, twists, and wrenches: the passive interpretation gives the representations of the *same* screw in the two frames linked by the transformations; the active interpretation moves a screw from an initial position to a different final position.

6.6.4 Pose twist transformation

For pose twists (Sect. 6.5), the situation is a bit different. Under a change of world reference frame, the three-vectors that make up the twist do *not* change, since the velocity reference point is independent of the world frame. Only the coordinates of the vectors change because the three-vectors are *projected* onto another reference frame. Hence, pose twists transform under a change of world reference frame from the initial frame $\{i\}$ to the final frame $\{f\}$ with the following *pose twist transformation matrix* ${}^i_f\boldsymbol{P}$:

$$\boxed{{}_f\mathbf{t} = {}^i_f\boldsymbol{P}\ {}_i\mathbf{t} = \begin{pmatrix} {}^i_f\boldsymbol{R} & \boldsymbol{0}_3 \\ \boldsymbol{0}_3 & {}^i_f\boldsymbol{R} \end{pmatrix}\ {}_i\mathbf{t}.} \tag{6.30}$$

${}^i_f\boldsymbol{P}$ has also always unit determinant.

Inverse of pose twist transformation The inverse of ${}^i_f\boldsymbol{P}$ is trivial:

$$({}^i_f\boldsymbol{P})^{-1} = \begin{pmatrix} {}^f_i\boldsymbol{R} & \boldsymbol{0}_3 \\ \boldsymbol{0}_3 & {}^f_i\boldsymbol{R} \end{pmatrix}. \tag{6.31}$$

Change of reference point on the moving body When the reference frame on the moving body changes, the origin of this reference frame changes too, and, since the translational velocity part of the pose twist is the velocity of this origin as seen from the world reference frame, this translational velocity three-vector changes also. The change is only due to a change in the moment arm of the angular velocity three-vector $\boldsymbol{\omega}$ on the screw axis; the translational velocity three-vector on the screw axis remains unchanged and hence also its coordinates with respect to the (unchanged) world reference frame. In total, the pose twist transforms under a change of reference point from the origin of the initial body-fixed reference frame $\{i\}$ to the origin of the final body-fixed reference frame $\{f\}$ as follows:

$$\mathbf{t}^f = {}^i_f\boldsymbol{M}\ \mathbf{t}^i = \begin{pmatrix} \boldsymbol{1}_3 & \boldsymbol{0}_3 \\ [\boldsymbol{p}^{f,i}] & \boldsymbol{1}_3 \end{pmatrix}\ \mathbf{t}^i. \tag{6.32}$$

The trailing superscript indicates the reference point for the pose twist. Equation (6.32) is valid with respect to any world reference frame in which the coordinates of the twists and vectors are expressed.

6.6.5 Impedance transformation

The coordinate transformations of stiffness, damping and inertia matrices (Sect. 3.10) under a change of reference frame is easily derived from the transformation of the twists and wrenches they act on. For example, the compliance matrix $\boldsymbol{C}$ works on a wrench $\mathbf{w}$ to produce an infinitesimal displacement twist $\mathbf{t}_\Delta$: $\mathbf{t}_\Delta = \boldsymbol{C}\,\mathbf{w}$. The transformation of this relation from an initial reference frame $\{i\}$ to a final reference frame $\{f\}$ is calculated as follows:

$$\begin{aligned} {}_f\mathbf{t}_\Delta &= {}^i_f\boldsymbol{S}\ {}_i\mathbf{t}_\Delta \\ &= {}^i_f\boldsymbol{S}\,({}_i\boldsymbol{C}\ {}_i\mathbf{w}) \\ &= {}_f\boldsymbol{C}\ {}_f\mathbf{w}. \end{aligned} \tag{6.33}$$

Hence

$$\boxed{{}_f\boldsymbol{C} = {}^i_f\boldsymbol{S}\,{}_i\boldsymbol{C}\;{}^i_f\boldsymbol{S}^{-1}.} \tag{6.34}$$

Similar reasonings apply to the stiffness and inertia matrices too. Note that Eq. (6.34) is a *similarity transformation*, Fact 34. Such transformations leave the *eigenvectors* and *eigenvalues* of the matrices unchanged, [22]. Of course, the coordinate description of the impedance matrices will change, but not the physical mapping they represent.

6.7 Second order time derivative of pose—Acceleration

Until now, only pose and velocity of rigid bodies in motion have been described, by means of homogeneous transformation matrices and twists. This Section discusses the second-order motion characteristics, i.e., acceleration. Recall that this means that we're looking for the minimum information required to calculate the acceleration of any point rigidly connected to a moving body.

6.7.1 Motor product—Derivative of screw along twist

It is not difficult to find an expression for the time derivative of a screw, *if* this screw is the twist generated by a revolute or prismatic joint fixed to a moving rigid body. Indeed, assume the body moves with a twist $\mathbf{t}^1 = ((\boldsymbol{\omega}^1)^T\,(\boldsymbol{v}^1)^T)^T$, and the twist generated by the joint is $\mathbf{t}^2 = ((\boldsymbol{\omega}^2)^T\,(\boldsymbol{v}^2)^T)^T$ with respect to the current world reference frame. After an infinitesimal time interval Δt, the body and the joint are transformed by the infinitesimal screw displacement $\boldsymbol{S}_\Delta(\mathbf{t}_\Delta = \Delta t\,\mathbf{t}^1)$. Hence, the time derivative of $\mathbf{t}^2$ is found as

$$\begin{aligned}\frac{d\mathbf{t}^2}{dt} &= \lim_{\Delta t\to 0}\frac{\boldsymbol{S}_\Delta(\Delta t\,\mathbf{t}^1)\mathbf{t}^2 - \mathbf{t}^2}{\Delta t}\\ &= \begin{pmatrix}[\boldsymbol{\omega}^1] & \boldsymbol{0}\\ [\boldsymbol{v}^1] & [\boldsymbol{\omega}^1]\end{pmatrix}\mathbf{t}^2 &(6.35)\\ &\triangleq \mathbf{t}^1\times\mathbf{t}^2. &(6.36)\end{aligned}$$

This last relationship is *motor product*, Sect. 4.5.3

6.7.2 Acceleration of rigid body points

At first sight, the definition of the *acceleration* of a rigid body is not difficult: just take the time derivative of the body's velocity twist, as is done for a moving *point*. However, this approach misses some important properties of a *rigid body* motion, i.e., those caused by the interaction of angular and linear motion components. To make this explicit, we start from the velocity of an arbitrary point attached to a moving rigid body, Eq. (6.7):

$${}_a\dot{\boldsymbol{p}} = [{}_a\boldsymbol{\omega}]\;{}_a\boldsymbol{p} + {}_a\dot{\boldsymbol{p}}^{a,b} - [{}_a\boldsymbol{\omega}]\;{}_a\boldsymbol{p}^{a,b}.$$

Taking the time derivative of both sides and using Eq. (6.8) gives

$$\begin{aligned}{}_a\ddot{\boldsymbol{p}} &= [{}_a\dot{\boldsymbol{\omega}}]\;{}_a\boldsymbol{p} + [{}_a\boldsymbol{\omega}]\;{}_a\dot{\boldsymbol{p}} + {}_a\ddot{\boldsymbol{p}}^{a,b} - [{}_a\dot{\boldsymbol{\omega}}]\;{}_a\boldsymbol{p}^{a,b} - [{}_a\boldsymbol{\omega}]\;{}_a\dot{\boldsymbol{p}}^{a,b}\\ &= [{}_a\dot{\boldsymbol{\omega}}]\;{}_a\boldsymbol{p} + [{}_a\boldsymbol{\omega}]\;[{}_a\boldsymbol{\omega}]\;{}_a\boldsymbol{p} + {}_a\ddot{\boldsymbol{p}}^{a,b} - [{}_a\dot{\boldsymbol{\omega}}]\;{}_a\boldsymbol{p}^{a,b} - [{}_a\boldsymbol{\omega}]\;[{}_a\boldsymbol{\omega}]\;{}_a\boldsymbol{p}^{a,b}\\ &= ([{}_a\dot{\boldsymbol{\omega}}] + [{}_a\boldsymbol{\omega}]\;[{}_a\boldsymbol{\omega}])\;{}_a\boldsymbol{p} + {}_a\boldsymbol{a}_0, &(6.37)\end{aligned}$$

with ${}_a\boldsymbol{a}_0 = \dot{\boldsymbol{v}}_0 + [{}_a\boldsymbol{\omega}]\ {}_a\boldsymbol{v}_0$, and ${}_a\boldsymbol{v}_0$ as in Eq. (6.9). These are, respectively, the acceleration and velocity of the point of the moving body that instantaneously coincides with the origin of $\{a\}$. $\dot{\boldsymbol{v}}_0$ is sometimes called the *tangential acceleration*; $[\boldsymbol{\omega}]\boldsymbol{v}_0$ is the *normal acceleration*; $[{}_a\boldsymbol{\omega}]\ [{}_a\boldsymbol{\omega}]_a\boldsymbol{p}$ is the *Coriolis acceleration*, [1, 8]. In general, the determinant of the coefficient matrix of $\boldsymbol{p}$ in Eq. (6.37) does *not* vanish, such that a point with instantaneous vanishing acceleration exists: solve for ${}_a\boldsymbol{p}$ from Eq. (6.37), with left-hand side equal to zero. This point is often called the *acceleration centre*, or *acceleration pole*, [2, 3, 10, 19, 21]. The matrix form of Eq. (6.37) is straightforwardly found from Eq. (6.8), [23]:

$$\begin{aligned}
\begin{pmatrix} {}_a\ddot{\boldsymbol{p}} \\ 0 \end{pmatrix} &= \frac{d}{dt}\left({}^b_a\dot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1}\right)\begin{pmatrix} {}_a\boldsymbol{p} \\ 1 \end{pmatrix} + {}^b_a\dot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1}\begin{pmatrix} {}_a\dot{\boldsymbol{p}} \\ 0 \end{pmatrix} \\
&= \left({}^b_a\ddot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1} + {}^b_a\dot{\boldsymbol{T}}\ \frac{d}{dt}({}^b_a\boldsymbol{T}^{-1}) + {}^b_a\dot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1}{}^b_a\dot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1}\right)\begin{pmatrix} {}_a\boldsymbol{p} \\ 1 \end{pmatrix} \\
&= \left({}^b_a\ddot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1} - {}^b_a\dot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1}{}^b_a\dot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1} + {}^b_a\dot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1}{}^b_a\dot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1}\right)\begin{pmatrix} {}_a\boldsymbol{p} \\ 1 \end{pmatrix}.
\end{aligned} \tag{6.38}$$

or

$$\boxed{\begin{pmatrix} {}_a\ddot{\boldsymbol{p}} \\ 0 \end{pmatrix} = {}^b_a\ddot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1}\begin{pmatrix} {}_a\boldsymbol{p} \\ 1 \end{pmatrix},} \quad \text{with} \quad {}^b_a\ddot{\boldsymbol{T}} = \begin{pmatrix} \left([\dot{\boldsymbol{\omega}}] + [\boldsymbol{\omega}][\boldsymbol{\omega}]\right){}^b_a\boldsymbol{R} & {}_a\ddot{\boldsymbol{p}}^{a,b} - [\boldsymbol{\omega}]_a\dot{\boldsymbol{p}}^{a,b} - [\dot{\boldsymbol{\omega}}]_a\dot{\boldsymbol{p}}^{a,b} \\ \boldsymbol{0}_{1\times 3} & 0 \end{pmatrix}. \tag{6.39}$$

Again, one gets a linear mapping from the coordinates of the point $\boldsymbol{p}$ to its acceleration. However, the angular velocity of the moving body enters non-linearly in this mapping.

Contrary to what was the case for the velocity analysis of the moving body (Sect. 6.5), the information contained in $\ddot{\boldsymbol{T}}\boldsymbol{T}^{-1}$ *cannot* be reduced to two three-vectors, Sect. 6.4: it contains linear and angular velocity three-vectors, as well as their time derivatives, so *four independent three-vectors* in total.

Fact-to-Remember 43 (Rigid body acceleration is not a screw vector)
The acceleration of a moving rigid body <u>cannot</u> be represented in one single six-vector.

6.7.3 Second-order screw axis

The literature on the application of screw theory in kinematics contains a representation of the acceleration of a rigid body that uses *two* six-vectors, [3, 21, 20]. These two six-vectors represent two *instantaneous screw axes*, Sect. 3.9: (i) the first order ISA that represents the velocity of the body, and (ii) the second order ISA that represents the velocity of the first ISA (Fig. 6.4). At each instant in time the moving body has, in general, a different ISA, and together these ISAs generate a ruled surface, called the *axode* (or *axoid*) of the motion, [4, p. 158], [12, 17, 19], [23, p. 240–243]. Two subsequent axodes have a common normal; this common normal is unique for a general motion, but it degenerates, for example, when the ISA doesn't change or moves parallel with itself. The common normal intersects the ISA in the so-called *central point*. Now, the motion of the ISA can be modelled by a translation along this common normal, plus a rotation about it. This combination of translation along, and rotation about, the same line is exactly what a screw axis is. These two ISAs are sufficient to model all 12 components of the moving body's acceleration:

1. The linear and angular velocity three-vectors of the body lie on the first order ISA. (This is a screw with 6 components.)

2. This same ISA also contains the angular acceleration $\boldsymbol{\alpha}$ and the linear acceleration $\boldsymbol{a}$ along the ISA itself. (One needs only 2 extra components for these two vectors, since one knows that they lie on the ISA.)

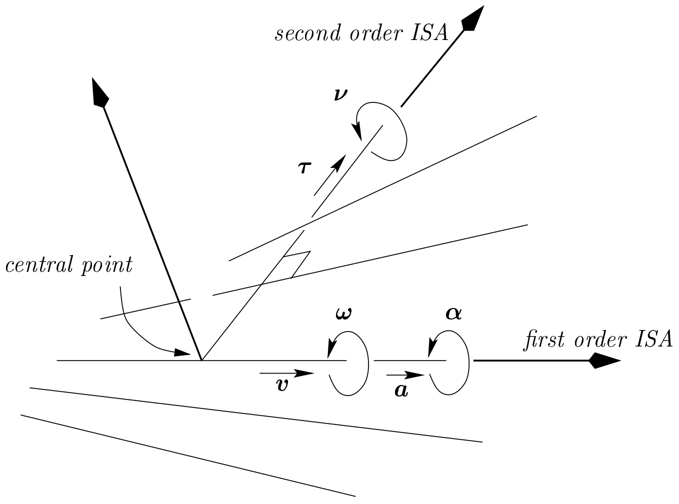


Figure 6.4: First order screw axes (ISA) for a general rigid body motion at different instants in time. The common normal between two subsequent ISAs is the second order screw axis.

3. The second order ISA contains the angular velocity $\boldsymbol{\nu}$ of the first order ISA, as well as the linear velocity $\boldsymbol{\tau}$ of the central point. (The second order ISA needs 2 parameters for its representation: one coordinate along the ISA, describing their intersection point, and one angle describing its orientation about the ISA; $\boldsymbol{\nu}$ and $\boldsymbol{\tau}$ need 2 more parameters, representing their magnitudes; their direction is already determined by the second order ISA.)

References for this Chapter

[1] J. Angeles. *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms*. Mechanical Engineering Series. Springer-Verlag, 1997.

[2] R. Beyer. *Technische Raumkinematik*. Springer Verlag, Berlin, Deutschland, 1963.

[3] E. H. Bokelberg, K. H. Hunt, and P. R. Ridley. Spatial motion—I. Points of inflection and the differential geometry of screws. *Mechanism and Machine Theory*, 27(1):1–15, 1992.

[4] A. Bottema and B. Roth. *Theoretical Kinematics*. Dover Books on Engineering. Dover Publications, Inc., Mineola, NY, 1990.

[5] H. Bruyninckx. Some invariance problems in robotics. Technical Report 91P04, Katholieke Universiteit Leuven, Dept. Mechanical Engineering, Belgium, 1991.

[6] J. Duffy. The fallacy of modern hybrid control theory that is based on "orthogonal complements" of twist and wrench spaces. *J. Robotic Systems*, 7(2):139–144, 1990.

[7] R. Featherstone. *Robot dynamics algorithms*. Kluwer Academic Publishers, Boston, MA, 1987.

[8] H. Goldstein. *Classical mechanics*. Addison-Wesley Series in Physics. Addison-Wesley, Reading, MA, 2nd edition, 1980.

[9] S. Helgason. *Differential geometry, Lie groups, and symmetric spaces*, volume 80 of *Pure and Applied Mathematics*. Academic Press, New York, NY, 1978.

[10] J. Hirschhorn. A graphical investigation of the acceleration pattern of a rigid body in three-dimensional motion. *Mechanism and Machine Theory*, 22(6):515–521, 1987.

[11] K. H. Hunt. Robot kinematics—A compact analytic inverse solution for velocities. *Trans. ASME J. Mech. Transm. Automation Design*, 109:42–49, 1987.

[12] K. H. Hunt. *Kinematic Geometry of Mechanisms*. Oxford Science Publications, Oxford, England, 2nd edition, 1990.

[13] H. Lipkin and J. Duffy. Hybrid twist and wrench control for a robotic manipulator. *Trans. ASME J. Mech. Transm. Automation Design*, 110:138–144, 1988.

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

[15] J. A. Parkin. Co-ordinate transformations of screws with applications to screw systems and finite twists. *Mechanism and Machine Theory*, 25(6):689–699, 1990.

[16] M. Renaud. *Contribution à l'étude de la modélisation et à la commande dynamiques des robots manipulateurs*. PhD thesis, Université Paul Sabatier, Toulouse, 1980.

[17] F. Reuleaux. *The kinematics of machinery*. MacMillan, London, England, 1876. Republished 1963 by Dover Publ. Co., New York.

[18] D. H. Sattinger and O. L. Weaver. *Lie groups and algebras with applications to physics, geometry, and mechanics*, volume 61 of *Applied Mathematical Sciences*. Springer-Verlag, New York, NY, 1986.

[19] M. Skreiner. A study of the geometry and the kinematics of instantaneous spatial motion. *Journal of Mechanisms*, 1:115–143, 1966.

[20] M. Skreiner. On the points of inflection in general spatial motion. *Journal of Mechanisms*, 2:429–433, 1967.

[21] H. J. Sommer, III. Determination of first and second order instant screw parameters from landmark trajectories. *Trans. ASME J. Mech. Design*, 114:274–282, 1992.

[22] G. Strang. *Introduction to applied mathematics*. Wellesley-Cambridge Press, Wellesley, MA, 1986.

[23] C. H. Suh and C. W. Radcliffe. *Kinematics and mechanisms design*. Wiley, New York, NY, 1978.

[24] R. von Mises. Motorrechnung, ein neues Hilfsmittel der Mechanik. *Zeitschrift für Angewandte Mathematik und Mechanik*, 4(2):155–181, 1924. English translation by E.J. Baker and K. Wohlhart, published by the Institute for Mechanics, University of Technology, Graz, Austria, 1996.

[25] L. S. Woo and F. Freudenstein. Application of line geometry to theoretical kinematics and the kinematic analysis of mechanical systems. *Journal of Mechanisms*, 5:417–460, 1970.

[26] M. S. C. Yuan and F. Freudenstein. Kinematic analysis of spatial mechanisms by means of screw coordinates. Part 1—Screw coordinates. *Trans. ASME J. Eng. Industry*, 93:61–66, 1971.

Chapter 7

Serial manipulator kinematics

7.1 Introduction

The *kinematics* of a robotic device studies the motion of the device, without considering the forces that cause this motion. The relationship between motion and force is the *dynamics* of robots, Chapt. 10. Kinematics are important, not only as an indispensable prerequisite for any dynamic description, but also for practical applications, such as motion planning and singularity analysis. This Chapter presents the *displacement* and *velocity* characteristics of *serial* kinematics chains. In this context, the major questions are: *"What are the relationships between, on the one hand, the positions and velocities of the robot joints, and, on the other hand, the position and velocity of the robot's end-effector?"* and *"What are the 'best' methods to calculate these kinematic relationships on line, i.e., during the execution of the motion?"* This text focusses on the first question. In addition, it also discusses the *statics* of serial manipulators, i.e., the static equilibrium between forces on the robot end-effector and on the joint axes. The duality between forces and velocities, Chapter 3, is extensively used in this Chapter (and even more in the following Chapter that deals with *parallel* manipulators). This integrated analysis of kinematics and statics for rigid bodies and robotic devices is sometimes given the name *kinetostatics.*

Fact-to-Remember 44 (Basic ideas of this Chapter)
The position and orientation of a robot's end-effector are derived from the joint positions by means of a geometric model of the robot arm. For serial robots, the mapping from joint positions to end-effector pose is easy, the inverse mapping is more difficult. Therefore, most industrial robots have special designs that reduce the complexity of the inverse mapping. The most popular designs involve a spherical wrist.

The key geometrical concepts used in this Chapter are (i) the pose representations (homogeneous transform, finite displacement twist) that describe relative *displacements* of two rigid bodies in the three-dimensional Euclidean space, (ii) the *screw*, in the form of twists and wrenches, that represents relative velocity of two rigid bodies, as well as generalised forces on a rigid body, and (iii) the reciprocity of screws.

The joint positions, velocities and forces form *coordinates* on, respectively, SE(3) (i.e., the pose of the end-effector), se(3) (i.e., the twist of the end-effector), and se*(3) (i.e., the forces acting on the end-effector). Many robots have gear boxes between the joints and the actuators that drive these joints. Hence, the position of the joint is in general different from the position of the motor. Typical gear ratios are in the range 1/10–1/500, with the motors making more revolutions that the joints.

"*Cartesian coordinate*"

Fact-to-Remember 45 (Coordinates)
Two natural coordinate systems are commonly used to describe the motion of an object manipulated by a serial robot: (i) the "Cartesian" coordinates on SE(3) and its tangent and co-tangent spaces, and (ii) the joint coordinates.

The previous Chapters discussed Cartesian coordinates; this Chapter discusses the relationships between joint space and Cartesian space coordinate systems when using a serial robot to move the object.

7.2 Serial robot designs

In its most general form, a serial robot design consists of a number of rigid links connected with joints. Simplicity considerations in manufacturing and control have led to robots with only revolute or prismatic joints and orthogonal, parallel and/or intersecting joint axes (instead of arbitrarily placed joint axes). In his 1968 Ph.D. thesis, [54], Donald L. Pieper (1941–) derived the first practically relevant result in this context:

Fact-to-Remember 46 (Closed-form inverse kinematics)
The inverse kinematics of serial manipulators with six revolute joints, and with three consecutive joints intersecting, can be solved in closed-form, i.e., analytically.

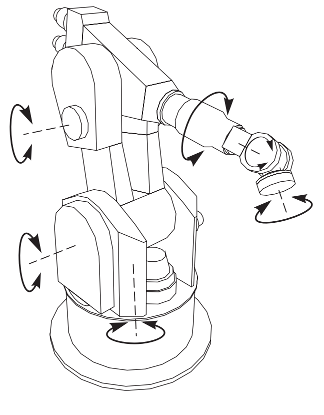


Figure 7.1: A *Kuka-160* serial robot.

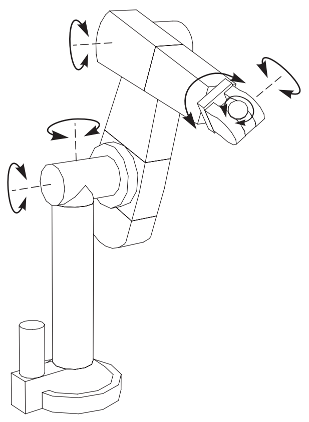


Figure 7.2: A *Staubli* (formerly *Unimation*) "PUMA" serial robot.

This result had a tremendous influence on the design of industrial robots: until 1974, when Cincinnati Milacron launched its T^3 robot (which has three consecutive *parallel* joints, i.e., intersecting at infinity, Fig. 7.3), all industrial manipulators had at least one prismatic joint [76] (see e.g., [71] for an impressively large catalogue) while since then, most industrial robots are *wrist-partitioned 6R* manipulators, such as shown in Figures 7.1 and 7.2. They have six revolute joints, and their last three joint axes intersect orthogonally, i.e., they form a *wrist* such as, for example, the *ZXZ* wrist in Fig. 5.5. This way, they can achieve any possible orientation. This construction leads to a *decoupling* of the position and orientation kinematics, for the forward as well as the inverse

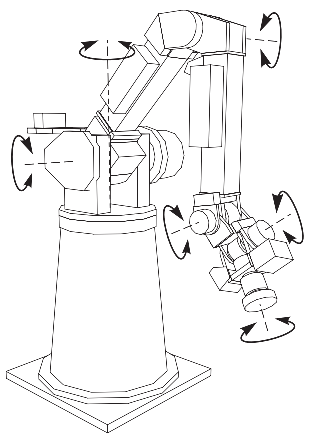


Figure 7.3: The *Cincinnati Milacron* T^3 serial robot.

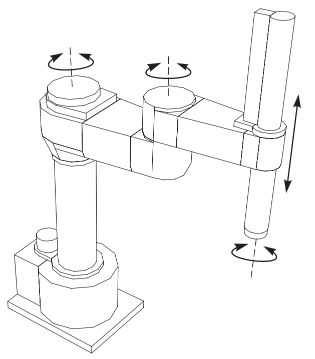


Figure 7.4: An *Adept SCARA* robot.

problems. For the three wrist joints, Section 5.2.8 already presented a solution; the remaining three joints are then found by solving a polynomial of, at most, fourth order, whatever their kinematic structure is, [54]. The extra structural simplifications (i.e., parallel or orthogonal axes) introduced in the serial robots of, for example, Figures 7.1 and 7.2, lead to even simpler solutions (Sect. 7.9.2). (Roughly speaking, each geometric constraint imposed on the kinematic structure simplifies the calculations.) The simplest kinematics are found in the *SCARA* robots (Selectively Compliant Assembly Robot Arm), Fig. 7.4. They have three vertical revolute joints, and one vertical prismatic joint at the end. These robots are mainly used for "pick-and-place" operations. In such a task, the robot must be stiff in the vertical direction (because it has to push things into other things) and a bit compliant in the horizontal plane, because of the imperfect relative positioning between the manipulated object and its counterpart on the assembly table. This desired selective compliance behaviour is intrinsic to the SCARA design; hence the name of this type of robots.

Design characteristics. The examples above illustrate the common design characteristics of commercial serial robot arms:

1. They are *anthropomorphic*, in the sense that they have a "shoulder," (first two joints) an "elbow," (third joint) and a "wrist" (last three joints). So, in total, they have the *six degrees of freedom* needed to put an object in an arbitrary position and orientation.
2. Almost all commercial serial robot arms have only *revolute* joints. Compared to prismatic joints, revolute joints are cheaper and give a larger dextrous workspace for the same robot volume.
3. They are very *heavy*, compared to the maximum load they can move without loosing their accuracy: their useful load to own-weight ratio is worse than 1/10! The robots are so heavy because the links must be stiff: deforming links cause position and orientation errors at the end-point.

Hybrid designs. A last industrially important class of "serial" robot arms are the *gantry* robots, Fig. 7.5. They have three prismatic joints to position the wrist, and three revolute joints for the wrist. Strictly speaking, a gantry robot is a combination of a *parallel XYZ* translation structure with a *serial* spherical wrist. The parallel construction is very stiff (cf. metal cutting machines) so that these robots are very accurate. In large industrial applications (such as welding of ship hulls or other large objects) a serial manipulator is often attached to a two

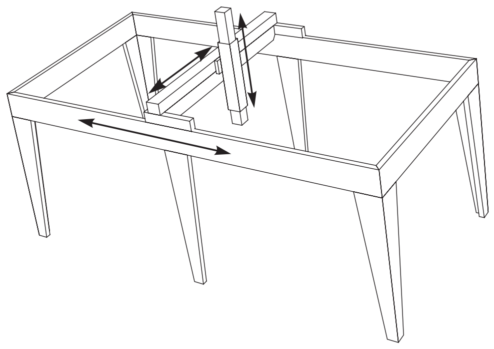


Figure 7.5: A gantry robot. (Only the first three prismatic degrees of freedom are shown.)

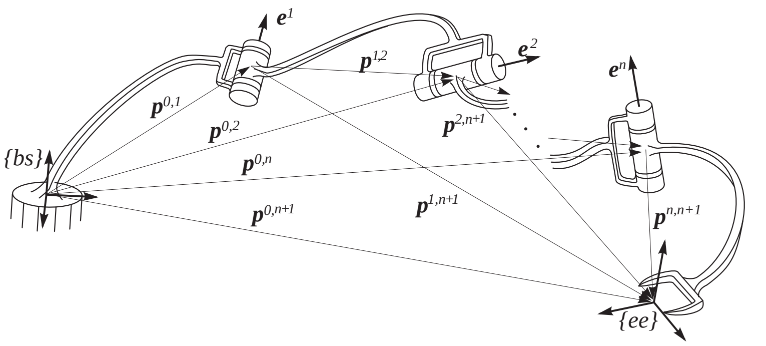


Figure 7.6: Notations used in the geometrical model of a serial kinematic chain.

or three degrees of freedom gantry structure, in order to combine the workspace and dexterity advantages of both kinematic structures.

Many other designs have been studied and implemented, but this text will stick to structures similar to the examples above, because:

Fact-to-Remember 47 (Decoupled kinematics of serial robots)
Simplicity of the forward and inverse position and velocity kinematics has always been one of the major design criteria for commercial manipulator arms. Hence, almost all of them have a very special kinematic structure that looks like either the SCARA design (Fig. 7.4), the gantry design (Fig. 7.5), or the 321 design (Fig. 7.9). These designs have efficient closed-form solutions because they allow for the <u>*decoupling*</u> *of the position and orientation kinematics. The geometric feature that generates this decoupling is the* <u>*intersection*</u> *of joint axes, Fact 46.*

7.3 Workspace

The *reachable workspace* of a robot's end-effector (or "*mounting plate*") is the manifold of reachable frames, i.e., a subset of SE(3). The *dextrous workspace* consists of the points of the reachable workspace where the robot can generate velocities that span the complete *tangent space* tangent space at that point, i.e., it can translate the manipulated object with three degrees of translation freedom, and rotate the object with three degrees of rotation freedom, [35, 52].

The relationships between joint space and Cartesian space coordinates of the object held by the robot are in general *multiple-valued*: the same pose can be reached by the serial arm in different ways, each with a different set of joint coordinates. Hence, the reachable workspace of the robot is divided in *configurations* (also called *assembly modes*), in which the kinematic relationships are *locally* one-to-one.

7.4 Link frame conventions

Coordinate representations of robotic devices must allow to represent the relative pose and velocity of two neighbouring links, as a function of the position and velocity of the joint connecting both links. This Chapter assumes that all joints and links are perfectly stiff, such that the kinematic model is purely geometrical, as in Fig. 7.6. The "base" frame $\{bs\}$ gets the index "0," and, for a manipulator with n joints, the end-effector frame has index "$n+1$." The direction vector $\boldsymbol{e}^i$ represents the positive direction of the ith joint axis. The position vector $\boldsymbol{p}^{i,j}$ connects the origin of link frame $\{i\}$ to the origin of link frame $\{j\}$. These link frames have one of their axes (usually the Z-axis) along the joint axis. Hence, the origin of these frames lies on the joint axis too.

The link closest to the base is sometimes called the *proximal link*; the link to which the end-effector is rigidly connected is the *distal* link.

7.4.1 Denavit-Hartenberg link frame convention

Joint axes are (directed) *lines*, and their representation needs minimally four parameters (Sect. 4.4.2). Figure 7.7 shows the four *Denavit-Hartenberg* parameters d, α, θ and h for the line Z^i, [15, 24], as well as the convention to define the frame $\{X^i, Y^i, Z^i\}$. The relative pose of $\{i\}$ with respect to $\{i-1\}$ is defined as follows.

1. The joint axis is the Z^i axis. This means that the joint coordinate $q^i = \alpha^i$ if joint i is revolute, and $q^i = h^i$ if the joint is prismatic. The positive sense of the axis corresponds to the positive sense of the joint position sensor.

2. The common normal between Z^i and Z^{i-1} determines the other axes of frame $\{i\}$:

   - The origin of frame $\{i\}$ is the intersection of Z^i and the common normal.
   - The X^i-axis lies along the common normal, with positive sense from the origin of $\{i-1\}$ to Z^i.

   If Z^i and Z^{i-1} are co-planar, the sense of X^i can be chosen arbitrarily. Also for the first (i.e., zeroth) frame, the X^1 direction is arbitrary.

3. The base frame $\{0\}$ and the end-effector frame $\{n+1\}$ cannot be chosen completely arbitrary: they must coincide with frames $\{1\}$ and $\{n\}$, respectively, when joints 1 and n are in their "zero" position. (This zero joint position can be chosen arbitrarily.) Often base and end-effector frames are placed at other locations than $\{0\}$ and $\{n+1\}$, because the structure of the kinematic chain suggests more "natural" positions for them; e.g., the end-effector frame is put at the end-point of the last link, or the base frame is put on the ground instead of on the first joint. However, if one wants to place them arbitrarily, one must use general pose transformations instead of pose transformations generated with DH parameters, since a DH transform allows for four independent parameters only.

Note that X^i is chosen to indicate a *fixed* direction with respect to the part of the joint that is fixed to the *previous* joint. That means that the joint value q^i can use X^i as "zero" reference: if the joint is revolute, the *direction* of X^i is the zero reference; if the joint is prismatic, the *position* of (the origin on) X^i is the zero reference.

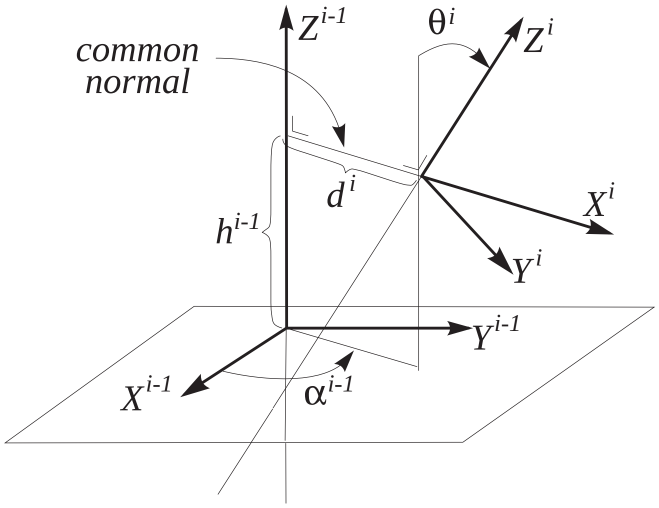


Figure 7.7: Frame definition in the Denavit-Hartenberg convention.

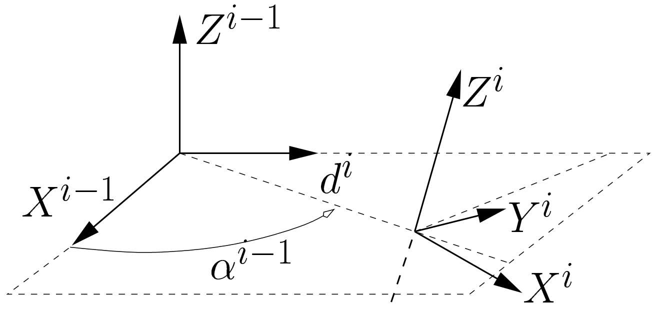


Figure 7.8: Frame definition in the Hayati-Roberts convention.

Link transformation matrix The homogeneous transformation matrix ${}_{i-1}^{\ \ i}\boldsymbol{T}$ representing the relative pose of two subsequent link frames $\{i-1\}$ and $\{i\}$ is straightforwardly derived from the DH link frame convention:

$$ {}_{i-1}^{\ \ i}\boldsymbol{T} = \boldsymbol{R}(Z,\alpha^{i-1})\,\boldsymbol{Tr}(Z,h^{i-1})\,\boldsymbol{Tr}(X,d^i)\,\boldsymbol{R}(X,\theta^i) \tag{7.1}$$

$$ = \begin{pmatrix} c_\alpha^{i-1} & -s_\alpha^{i-1} & 0 & 0 \\ s_\alpha^{i-1} & c_\alpha^{i-1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & h^{i-1} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & d^i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c_\theta^i & -s_\theta^i & 0 \\ 0 & s_\theta^i & c_\theta^i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{7.2}$$

or

$$ \boxed{ {}_{i-1}^{\ \ i}\boldsymbol{T} = \begin{pmatrix} c_\alpha^{i-1} & -s_\alpha^{i-1}c_\theta^i & s_\alpha^{i-1}s_\theta^i & d^i c_\alpha^{i-1} \\ s_\alpha^{i-1} & c_\alpha^{i-1}c_\theta^i & -c_\alpha^{i-1}s_\theta^i & 0 \\ 0 & s_\theta^i & c_\theta^i & h^{i-1} \\ 0 & 0 & 0 & 1 \end{pmatrix} }. \tag{7.3}$$

"$\boldsymbol{R}(Z,\alpha)$" represents the homogeneous transformation matrix that corresponds to the rotation about the Z-axis, over an angle α (Sect. 5.2.2); "$\boldsymbol{Tr}(Z,h)$" represents the homogeneous transformation matrix that corresponds to the translation along Z over a distance h, etc. Note that, in general, the *inverse* transformation (from a given *arbitrary* $\boldsymbol{T}$ to a set of DH parameters α, h, d and θ) does not exist: the four DH parameters cannot represent the six degrees of freedom of choosing an arbitrary reference frame.

Not all references use the same link frame conventions as outlined above! So be careful when using sets of DH parameters, and make sure to document all background information about how the parameters are defined.

7.4.2 Hayati-Roberts link frame convention

This section uses the Hayati-Roberts (HR) line convention (Sect. 4.4.4) to represent subsequent links. It is a substitute for the DH convention in the case of (nearly) parallel lines. Similarly to the DH case, there is no unique HR convention! Figure 4.3 showed one possible definition; Figure fig-HR-frames shows another. Two Euler angles β_i and γ_i, needed to represent the direction of the Z^i axis, are not shown. Since the HR convention is supposed to work when the two joint axes are almost parallel, Roll and Pitch angles are appropriate choices for β_i and γ_i (Sect. 5.3.3). The following four translations and rotations map the frame on the first joint axis onto the frame on the second joint axis:

$$
{}_{i-1}^{\ \ i}\boldsymbol{T} = \boldsymbol{R}(Z,\alpha^{i-1})\,\boldsymbol{Tr}(X,d^{i-1})\,\boldsymbol{R}(Y,\gamma^{i})\boldsymbol{R}(X,\beta^{i}). \tag{7.4}
$$

The "common normal" is not used in the HR convention (as it is in the DH convention). The origin of frame $\{i\}$ is *chosen* to lie in the $X^{i-1}Y^{i-1}$-plane.

Although the Hayati-Roberts convention avoids the coordinate singularity for parallel joint axes, it has itself a singularity when joint i is parallel to either X^{i-1} or Y^{i-1} (intersection with XY plane not defined), *or* when joint i intersects the origin of frame $\{i-1\}$ (α^{i-1} not defined), [6, 61].

Fact-to-Remember 48 (Link frame conventions)
The Denavit-Hartenberg and Hayati-Roberts link frame conventions define a homogeneous transformation matrix as a function of <u>*four*</u> *geometric parameters.*

7.5 321 kinematic structure

Because all-revolute joint manipulators have good workspace properties, and because a sequence of three intersecting joint axes introduces significant simplifications in the kinematic algorithms (Fact 46), most commercial robot arms now have a kinematic structure as shown in Fig. 7.9. (Vic Scheinman of Stanford University was, to the best of the authors' knowledge, the first to come up with this design, but he did not write it up in any readily accessible publications. . .) The design is an example of a *6R wrist-partitioned* manipulator: the last three joint axes intersect orthogonally at one point. Moreover, the second and third joints are parallel, and orthogonal to the first joint. These facts motivate the name of "*321*" robot arm: the *three* wrist joints intersect; the *two* shoulder and elbow joints are parallel, hence they intersect at infinity; the *first* joint orthogonally intersects the first shoulder joint.

The 321 can use a link frame transformation convention that is much simpler [21] than the Denavit-Hartenberg or Hayati-Roberts conventions because its geometry is determined by orthogonal and parallel joint axes, and by only four link lengths l_1, l_2, l_3 and l_6 (the wrist link lengths l_4 and l_5 are zero). The reference frames are all parallel when the robot is in its fully upright configuration. This configuration is also the *kinematic zero* position, i.e., all joint angles are *defined* to be zero in this position. The six joints are *defined* to rotate in positive sense about, respectively, the $+Z^1$, $-X^2$, $-X^3$, $+Z^4$, $-X^5$, and $+Z^6$ axes, such that positive joint angles make the robot "bend forward" from its kinematic zero position. Many industrial robots have a 321 kinematic structure, but it is possible that the manufacturers defined different zero positions and positive rotation directions for some joints. These differences are easily compensated by (constant) joint position offsets and joint position sign reversals.

321 kinematic structure with offsets. Many other industrial robots, such as for example the PUMA (Fig 7.2), have a kinematic structure that deviates a little bit from the 321 structure of Figure 7.9, [13, 65, 76]:

1. *Shoulder offset*: frame $\{3\}$ in Figure 7.9 is shifted a bit along the X-axis. This brings the elbow off-centre with respect to the line of joint 1.

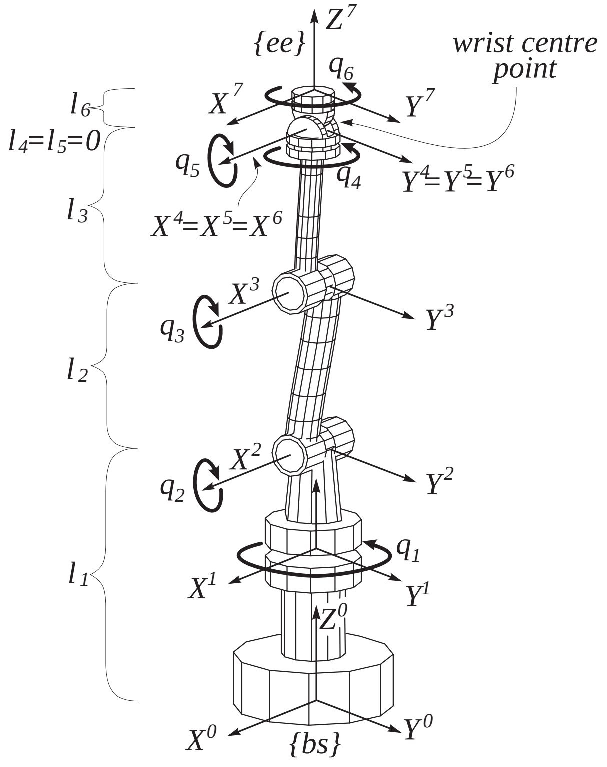


Figure 7.9: 321 kinematic structure in the "zero" position: all link frames are parallel and all origins lie on the same line.

2. *Elbow offset*: frame {4} in Figure 7.9 is shifted a bit along the Y-axis. This brings the wrist centre point off-centre with respect to the forearm.

The reasons for the offsets will become clear after the Section on singularities (Sect. 7.13): the offsets move the singular positions of the robot away from places in the workspace where they are likely to cause problems.

7.6 Forward position kinematics

The forward position kinematics (FPK) solves the following problem: *Given the joint positions* $\boldsymbol{q} = (q_1 \ \dots \ q_n)^T$, *what is the corresponding end-effector pose?* The solution is always unique: one given joint position vector always corresponds to only one single end-effector pose. The FK problem is not difficult to solve, even for a completely arbitrary serial kinematic structure.

7.6.1 General FPK: link transform algorithm

The easiest approach to calculate the FPK is to apply the composition formula (6.4) for homogeneous transformation matrices from the end-effector frame $\{ee\}$ to the base frame $\{bs\}$:

$$\boxed{{}^{ee}_{bs}\boldsymbol{T} = {}^{0}_{bs}\boldsymbol{T}\ {}^{1}_{0}\boldsymbol{T}(q_1)\ {}^{2}_{1}\boldsymbol{T}(q_2)\dots\ {}^{n}_{n-1}\boldsymbol{T}(q_n)\ {}^{ee}_{n}\boldsymbol{T}.} \tag{7.5}$$

Each link transform can be found, for example, from the Denavit-Hartenberg link frame definition. This approach works for *any* serial robot, with any number of revolute and/or prismatic joints. The resulting mapping from joint angles to end-effector pose is *nonlinear* in the joint angles. When implementing this procedure in a computer program, one should, of course, not code the complete matrix multiplications of Eq. (7.5), since (i) the last rows of the homogeneous transformation matrices are mostly zeros, and (ii) many robots have a kinematic structure that generates many more zeros in the rest of the matrices too.

7.6.2 Closed-form FPK for 321 structure

Serial manipulators of the 321 type allow for the decoupling of the robot kinematics at the wrist, for position as well as velocity, and for the forward as well as the inverse problems. This decoupling follows from the fact that the wrist has three intersecting revolute joints, and hence *any* orientation can be achieved by the wrist alone. This Section (and all the following Sections that threat closed-form 321 kinematics) starts by "splitting" the manipulator at the *wrist centre point* (reference frame $\{4\}$ in Fig. 7.9). This has the following advantages:

1. The position and linear velocity of the wrist centre point are completely determined by the first three joint positions and velocities.

2. The relative orientation and angular velocity of the last wrist frame $\{6\}$ with respect to the first wrist frame $\{4\}$ are completely determined by the three wrist joints.

3. The relative pose of the end-effector frame $\{7\}$ with respect to the last wrist frame $\{6\}$ is simply a *constant* translation along the Z^6-axis. A similar relationship holds between the frames on the base and on the first link.

The following procedure applies this approach to the forward position kinematics:

Closed-form FPK

Step 1 Section 5.3.2 has already calculated the closed-form forward *orientation* kinematics of the wrist, since this wrist is an instantiation of a ZXZ Euler angle set, upto the small difference that the positive sense of the rotation of the second wrist joint is about the $-X$ axis. The resulting homogeneous transformation matrix from $\{4\}$ to $\{6\}$ (of which Eq. (5.27) contains the rotation part) is repeated here, with the angles α, β and γ replaced by the joint angles q_4, q_5 and q_6, and taking into account the sign difference for q_5:

$$ {}^{6}_{4}\boldsymbol{T} = \begin{pmatrix} {}^{6}_{4}\boldsymbol{R} & \boldsymbol{0}_{3\times1} \\ \boldsymbol{0}_{1\times3} & 1 \end{pmatrix} = \begin{pmatrix} c_6c_4 - s_6c_5s_4 & -s_6c_4 - c_6c_5s_4 & -s_5s_4 & 0 \\ c_6s_4 + s_6c_5c_4 & -s_6s_4 + c_6c_5c_4 & s_5c_4 & 0 \\ -s_6s_5 & -c_6s_5 & c_5 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{7.6} $$

Step 2 The pose of the wrist reference frame $\{4\}$ with respect to the base reference frame $\{0\} = \{bs\}$ of the robot (i.e., ${}^{4}_{0}\boldsymbol{T}$) is determined by the first three joints. q_2 and q_3 are parallel, so they move the centre of the wrist in a plane, whose rotation about the Z axis of the base reference frame $\{bs\}$ is determined by q_1 only. q_2 and q_3 move the wrist to a vertical height d^v above the *shoulder* reference frame $\{2\}$ (i.e., $d^v + l_1$ above X^0Y^0) and to a horizontal distance d^h *in the arm plane*, i.e., the YZ-plane of $\{2\}$ (Fig. 7.10):

$$ \boxed{d^v = c_2l_2 + c_{23}l_3, \qquad d^h = s_2l_2 + s_{23}l_3,} \tag{7.7} $$

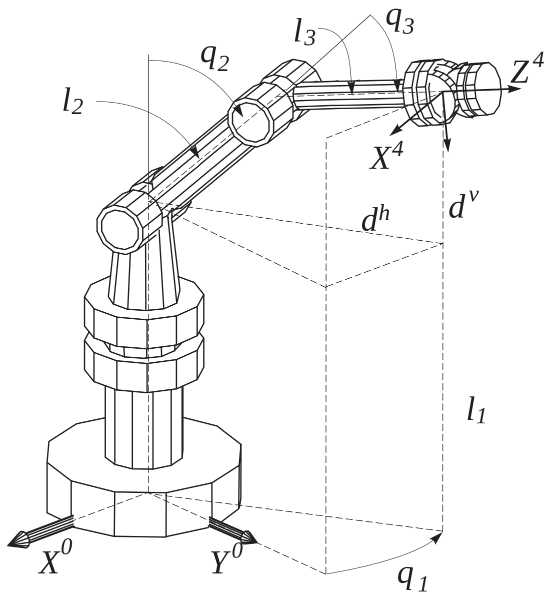


Figure 7.10: Kinematics of first three joints of the 321 manipulator (Fig. 7.9).

with $c_2 = \cos(q_2)$, $c_{23} = \cos(q_2 + q_3)$, etc. The contribution of the first three joints to the total orientation matrix consists of a rotation about Z_1, over an angle q_1, followed by a rotation about the *moved* X_2-axis, over an angle $q_2 + q_3$. Hence (Sect. 5.2.8):

$$
{}^4_0\boldsymbol{R} = \begin{pmatrix} c_1 & -s_1 & 0 \\ s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{23} & -s_{23} \\ 0 & s_{23} & c_{23} \end{pmatrix} = \begin{pmatrix} c_1 & -s_1 c_{23} & s_1 s_{23} \\ s_1 & c_1 c_{23} & -c_1 s_{23} \\ 0 & s_{23} & c_{23} \end{pmatrix}. \tag{7.8}
$$

Step 3 The pose of the end-effector reference frame $\{7\} = \{ee\}$ with respect to the last wrist reference frame $\{6\}$ (i.e., ${}^7_6\boldsymbol{T}$) corresponds to a translation along Z_6 over a distance l_6:

$$
{}^7_6\boldsymbol{T} = \begin{pmatrix} {}^7_6\boldsymbol{R} & l_6 \boldsymbol{e}_z \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_6 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{7.9}
$$

Step 4 Hence, the total orientation ${}^{ee}_{bs}\boldsymbol{R}$ follows from Eqs. (7.6), (7.8), and (7.9):

$$
\boxed{{}^{ee}_{bs}\boldsymbol{R} = {}^7_0\boldsymbol{R} = {}^4_0\boldsymbol{R}\; {}^6_4\boldsymbol{R}\; {}^7_6\boldsymbol{R}.} \tag{7.10}
$$

Step 5 The position of the wrist centre (i.e., the origin of $\{4\}$ with respect to the base $\{0\}$) is

$$
{}_{bs}\boldsymbol{p}^{wr} = {}_0\boldsymbol{p}^{wr} = \begin{pmatrix} c_1 d^h \\ s_1 d^h \\ l_1 + d^v \end{pmatrix}, \tag{7.11}
$$

and the position of the end-effector (i.e., the origin of $\{ee\}$ with respect to the base $\{0\}$) is

$$\boxed{{}_{bs}\boldsymbol{p}^{ee} = {}_{bs}\boldsymbol{p}^{wr} + {}^{ee}_{bs}\boldsymbol{R}\,(0\,0\,l_6)^T.} \tag{7.12}$$

Step 6 Equations (7.10) and (7.12) yield the final result:

$${}^{ee}_{bs}\boldsymbol{T} = \begin{pmatrix} {}^{ee}_{bs}\boldsymbol{R} & {}_{bs}\boldsymbol{p}^{ee} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix}. \tag{7.13}$$

7.7 Accuracy, repeatability, and calibration

Finding the pose of the end-effector given the actual joint positions relies on a mathematical idealization: in reality, the mathematical model is not 100% accurate (due to manufacturing tolerances) *and* the joint positions are not measured with infinite accuracy. This means that the real pose differs from the modelled pose. The smaller this difference, the better the *absolute (positioning) accuracy* of the robot, i.e., the mean difference between the actual pose and the pose calculated from the mathematical model. Absolute accuracy, however, is not the only important factor: in most industrial applications, robots are programmed *on line*, i.e, a human operator moves the end-effector to the desired pose and then stores the current values of the joint positions in the robot's electronic memory. This way, the absolute accuracy is not relevant, but rather the *repeatability* of the robot, i.e., the (mean) difference between the *actual* poses attained by the robot in subsequent (identical) motions to the same *desired* pose, whose corresponding joint values have been stored in memory.

Fact-to-Remember 49 (Accuracy—Repeatability)
The robot's repeatability is much better than its absolute accuracy, typically an order of magnitude.

For good industrial robots, the repeatability is of the order of 0.1mm. This is the *static* repeatability, i.e., the robot moves to the desired pose, and comes to a halt while the robot controller has sufficient time to make the robot reach this pose as accurately as possible.

Off-line programming and calibration More and more robots are programmed *off line*. This means that CAD (Computer Aided Design) drawings of the robot and its environment are used to (i) first interactively program the robot task on a graphical workstation until the desired functionality is reached, and (ii) then download the final task program to the robot work-cell. This approach has the advantage that it does not occupy the work-cell during the programming phase; its disadvantage is that it applies only to workcells in which the robots have a (very) high absolute accuracy, *and* the robot's environment is known with the same accuracy. Since it is expensive to build robots that correspond exactly to their nominal geometrical models, the practical solution to the absolute accuracy problem is to *calibrate* the robot, i.e., to adapt the geometrical model to the real kinematic structure of the robot, before bringing the robot in operation. ("Intelligent" robots follow an alternative approach: they use sensors to detect the errors *on line* and adapt the robot task accordingly.) A typical calibration procedure looks like this, [6, 25, 46, 68, 62]:

Calibration algorithm

Step 1 (Error model). One starts from the nominal geometric robot model, and adds a set $\{P\}$ of n *error parameters*. These parameters model the possible geometrical differences between the nominal and real kinematic structures. Of course, such an error model is a practical trade-off between (i) accuracy, and (ii) complexity. Common error parameters are offsets on the joint positions, joint axis line parameters, and base and end-effector frames. For example, $\Delta\alpha, \Delta h, \Delta\theta, \Delta d$ in the Denavit-Hartenberg link frame convention.

Step 2 (Data collection). The robot is moved to a large set of N different poses where its end-effector homogeneous transform ${}^{ee}_{bs}\boldsymbol{T}_m(\boldsymbol{q}_i), i = 1, \dots, N$ is calculated from the measured joint values and the nominal kinematic model. The number N of sampled poses is much larger than the number n of error parameters. The real end-effector and/or link frame poses ${}^{ee}_{bs}\boldsymbol{T}(\boldsymbol{q}_i, P)$ are measured with an accurate 3D measurement device (e.g., based on triangulation with laser or visual pointing systems), [6].

Step 3 (Parameter fitting). The real poses are expressed as a Taylor series in the error parameters $P_j, j = 1, \dots, n$:

$$
{}^{ee}_{bs}\boldsymbol{T}(\boldsymbol{q}_i, P) = {}^{ee}_{bs}\boldsymbol{T}(\boldsymbol{q}_i, 0) + \sum_{j=1}^{n} \left\{ \partial \left({}^{ee}_{bs}\boldsymbol{T}(\boldsymbol{q}_i, P) \right) / \partial P_j \right\} \; P_j + \mathcal{O}(P^2). \tag{7.14}
$$

The first term in this series is the pose ${}^{ee}_{bs}\boldsymbol{T}_m(\boldsymbol{q}_i)$ derived from the model. Taking only the first and second terms into account yields an overdetermined set of linear equations in the P_j. These P_j can then be fitted to the collected data in a "least-squares" sense. This means that the "distance" between the collected poses and the predictions made by the corrected model is minimal. Recall (Fact 7) that no unique distance function for poses exists. In principle, this fact would not influence the calibration result, since one tries to make the distance zero, and a zero distance *is* defined unambiguously for any *non-degenerate* distance function. However, the distance is never exactly zero, due to measurement noise and/or an incomplete error parameter set.

Step 4 (Model correction). With the error estimates obtained in the previous step, one adapts the geometric model of the robot.

The calibration procedure requires (i) the robot to be taken off line for a significant period of time, and (ii) expensive external measurement devices. However, once calibrated, the adapted geometric model does not vary much anymore over time. In practice, robot calibration often yields good absolute accuracy, but in a limited subset of the robot's workspace only. Note also that, due to the presence of the error parameters, a calibrated robot always has a *general* kinematic structure, even if its nominal model is of the 321 type. Hence, calibrated robots definitely need the *numerical* kinematic procedures described in this Chapter.

7.8 Forward velocity kinematics

The forward velocity kinematics (FVK) solves the following problem: *Given the vectors of joint positions* $\boldsymbol{q} = (q_1 \;\dots\; q_n)^T$ *and joint velocities* $\dot{\boldsymbol{q}} = (\dot{q}_1 \;\dots\; \dot{q}_n)^T$*, what is the resulting end-effector twist* $\mathbf{t}^{ee}$*?* The solution is always unique: one given set of joint positions and joint velocities always corresponds to only one single end-effector twist.

7.8.1 The Jacobian matrix

The relation between joint positions $\boldsymbol{q}$ and end-effector pose ${}^{T}\boldsymbol{T}$ is nonlinear, but the relationship between the joint velocities $\dot{\boldsymbol{q}}$ and the end-effector twist $\mathbf{t}^{ee}$ is *linear*: if one drives a joint twice as fast, the end-effector will move twice as fast too. (This linearity property corresponds to the fact that the tangent space se(3) is a vector space.) Hence, the linear relationship is *represented* by a *matrix*:

$$\boxed{\underset{6\times 1}{{}_{bs}\mathbf{t}^{ee}} = \underset{6\times n}{{}_{bs}\boldsymbol{J}(\boldsymbol{q})}\ \underset{n\times 1}{\dot{\boldsymbol{q}}}.} \tag{7.15}$$

The matrix ${}_{bs}\boldsymbol{J}(\boldsymbol{q})$ is called the *Jacobian matrix*, or *Jacobian* for short, with respect to the reference frame $\{bs\}$. It was introduced by Withney, [74] (see [54] for an earlier similar coordinate description that doesn't use the name "Jacobian"). The terminology is in accordance with the "Jacobian matrix" as defined in classical mathematical analysis, (i.e., the matrix of partial derivatives of a function, [8, 11, 43, 60, 66]) named after the Prussian mathematician Karl Gustav Jacob Jacobi (1804–1851). Note that the matrix of the linear mapping depends itself nonlinearly on the joint positions $\boldsymbol{q}$. One most often omits the explicit mention of $\boldsymbol{J}$'s dependence on the joint positions $\boldsymbol{q}$. Note that the mapping from joint velocities to end-effector motion is unique, but that different Jacobian *matrices* (i.e., coordinate *representations*) exist, depending on (i) whether the twist on the left-hand side of Eq. (7.15) is a screw twist, a pose twist or a body-fixed twist (Sect. 6.5), and (ii) the reference frame $\{bs\}$ with respect to which the end-effector twist $\mathbf{t}^{ee}$ is expressed.

Fact-to-Remember 50 (Physical interpretation of Jacobian matrix)
The ith column of the Jacobian matrix is the end-effector twist generated by a unit velocity applied at the ith joint, and zero velocities at the other joints.
The Jacobian matrix is a basis for the vector space of all possible end-effector twists; hence, each column of the Jacobian is sometimes called a partial twist, [45].

The twist interpretation of the Jacobian implies that the joint rates $\dot{\boldsymbol{q}}$ are *dimensionless* coordinates.

Analytical Jacobian $\boldsymbol{J}$ in Eq. (7.15) is *not* a real mathematical Jacobian, since the angular velocity three-vector $\boldsymbol{\omega}$ is not the time derivative of any three-vector orientation representation, Sect. 5.3.6. Nevertheless, the time derivative of a forward position kinematic function $\mathbf{t}_d = f(\boldsymbol{q})$ is well-defined:

$$\frac{d\mathbf{t}_d}{dt} = \sum_{i=1}^{n} \frac{\partial f(\boldsymbol{q})}{\partial q_i}\frac{\partial q_i}{\partial t} \triangleq \bar{\boldsymbol{J}}\dot{\boldsymbol{q}}. \tag{7.16}$$

The angular coordinates of the finite displacement twist $\mathbf{t}_d$ are a set of three Euler angles. Chapter 5 has shown that the time derivatives of these Euler angles are related to the angular velocity three-vector by means of *integrating factors*. Hence, the difference between the Jacobian in Eq. (7.15) and the matrix of partial derivatives $\bar{\boldsymbol{J}} = \partial f(\boldsymbol{q})/\partial q_i$ in Eq. (7.16) are these integrating factors. $\bar{\boldsymbol{J}}$ in Eq. (7.16) is sometimes called the *analytical Jacobian*, [23, 62, 32], when it is necessary to distinguish it from the *twist Jacobian* $\boldsymbol{J}$ in Eq. (7.15).

7.8.2 General FVK: velocity recursion

The previous Section *defines* the Jacobian matrix; this Section explains how *to calculate* it, starting from known joint positions and velocities. The following procedure works for any serial structure with an arbitrary number of n joints [51]. The basic idea is to perform an *outward recursion* (or "sweep"): one starts with the twist generated by the joint closest to the base, then transforms this twist to the second joint, adds the twist generated by this joint, transforms it to the third joint, etc.

Numerical FVK

Step 0 Initialization. The twist of the "zeroth" joint in the base reference frame $\{0\}$ is always zero:

$$i = 0, \quad \text{and} \quad {}_0\mathbf{t}^0 = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{pmatrix}. \tag{7.17}$$

Step 1 Recursion $i \rightarrow i+1$, until $i = n$:

Step 1.1 Transformation of the twist ${}_i\mathbf{t}^i$ to the next joint:

$${}_{i+1}\mathbf{t}^i = {}_{i+1}^{\;\;i}\boldsymbol{S}\ {}_i\mathbf{t}^i, \tag{7.18}$$

where the screw transformation matrix ${}_{i+1}^{\;\;i}\boldsymbol{S}$ is constructed from the (known) link transform ${}_{i+1}^{\;\;i}\boldsymbol{T}$ as described in Eq. (6.25).

Step 1.2 Add contribution of joint $i+1$:

$${}_{i+1}\mathbf{t}^{i+1} = {}_{i+1}\mathbf{t}^i \ + \ {}_{i+1}\boldsymbol{J}_{i+1}\ \dot{q}_{i+1}. \tag{7.19}$$

The Jacobian column ${}_{i+1}\boldsymbol{J}_{i+1}$ equals $(0\ 0\ 1\ 0\ 0\ 0)^T$ for a revolute joint, and $(0\ 0\ 0\ 0\ 0\ 1)^T$ for a prismatic joint, since the local Z-axis is defined to lie along the joint axis.

The result of the recursion is ${}_{n+1}\mathbf{t}^{n+1} = {}_{ee}\mathbf{t}^{ee}$, the total end-effector twist expressed in the end-effector frame $\{ee\}$.

Step 2 Transformation to the world frame $\{w\}$ gives:

$${}_w\mathbf{t}^{ee} = {}_w^{ee}\boldsymbol{S}\ {}_{ee}\mathbf{t}^{ee}. \tag{7.20}$$

The recursive procedure above also finds the Jacobian matrix: the second term in each recursion through Step 1.2 yields, for $\dot{q}_{i+1} = 1$, a new column of the Jacobian matrix, expressed in the local joint reference frame. Applying all subsequent frame transformations to this new Jacobian column results in its representation with respect to the world reference frame:

$${}_w\boldsymbol{J}_i = {}_1^w\boldsymbol{S}\ {}_0^1\boldsymbol{S}\ {}_1^2\boldsymbol{S}\ \dots\ {}_{i-1}^{\;\;i}\boldsymbol{S}\ {}_i\boldsymbol{J}_i.$$

Variations on this FVK algorithm have appeared in the literature, differing only in implementation details to make the execution of the algorithm more efficient.

7.8.3 Closed-form FVK for 321 structure

For the 321 kinematic structure, more efficient closed-form solutions exist, [21, 40, 57, 58, 69] and [29]. The approach of the last reference is especially instructive, since it maximally exploits geometric insight. The wrist centre frame $\{4\}$ of the 321 kinematic structure is the best choice as world frame, because it allows to solve the FVK *by inspection*, as the next paragraphs will show. (The Jacobian expressed in the wrist centre frame is sometimes called the "midframe" Jacobian, [17].)

Closed-form FVK

Step 1 The wrist is of the *ZXZ* type (Sect. 5.2.8). The angular velocity generated by the fourth joint lies along the Z^4-axis. The angular velocity generated by the fifth joint lies along the X^5-axis, that is found by rotating the X^4-axis about Z^4 over an angle q_4. And the angular velocity generated by the sixth joint lies along the Z^6-axis, whose orientation with respect to $\{4\}$ is found in the last column of Eq. (7.6). In total, this yields

$$ {}_4\boldsymbol{J}_{456} = \begin{pmatrix} {}_4\boldsymbol{J}_4 & {}_4\boldsymbol{J}_5 & {}_4\boldsymbol{J}_6 \end{pmatrix} = \begin{pmatrix} 0 & c_4 & -s_5 s_4 \\ 0 & s_4 & s_5 c_4 \\ 1 & 0 & c_5 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{7.21} $$

Step 2 The twists generated by joints $1, 2$ and 3 are pure rotations too, but they cause translational velocities at the wrist centre point due to the non-zero lever arms between the joints and the wrist centre point. These moments arms are ${}_4\boldsymbol{p}^{i,4}$, for $i = 1, 2, 3$, i.e., the position vectors from the three joints to the wrist centre point. Hence, inspection of Figure 7.10 yields

$$ {}_4\boldsymbol{J}_{123} = \begin{pmatrix} {}_4\boldsymbol{J}_1 & {}_4\boldsymbol{J}_2 & {}_4\boldsymbol{J}_3 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ -s_{23} & 0 & 0 \\ c_{23} & 0 & 0 \\ -d^h & 0 & 0 \\ 0 & l_2 c_3 + l_3 & l_3 \\ 0 & l_2 s_3 & 0 \end{pmatrix}. \tag{7.22} $$

Step 3 In order to obtain twists with the base frame as origin, it suffices to pre-multiply ${}_4\boldsymbol{J} = \begin{pmatrix} {}_4\boldsymbol{J}_{123} & {}_4\boldsymbol{J}_{456} \end{pmatrix}$ by the screw transformation matrix ${}^{4}_{bs}\boldsymbol{S}$:

$$ {}_{bs}\boldsymbol{J} = {}^{4}_{bs}\boldsymbol{S}\ {}_4\boldsymbol{J}. \tag{7.23} $$

${}^{4}_{bs}\boldsymbol{S}$ is straightforwardly derived from the solution to the forward position kinematics of the robot (Sect. 7.6.2).

The motivation for choosing the wrist centre frame as reference frame is illustrated by the fact that the Jacobian expressed in this frame has a zero 3×3 submatrix.

Later Sections will need the value of the determinant of the Jacobian matrix. Also here, the advantage of the midframe Jacobian ${}_4\boldsymbol{J}$ appears: it has a zero 3×3 submatrix, which enormously simplifies the calculation of the determinant:

$$ \begin{aligned} \det({}_4\boldsymbol{J}) &= \det \begin{pmatrix} -d^h & 0 & 0 \\ 0 & l_2 c_3 + l_3 & l_3 \\ 0 & l_2 s_3 & 0 \end{pmatrix} \det \begin{pmatrix} 0 & c_4 & -s_5 s_4 \\ 0 & s_4 & s_5 c_4 \\ 1 & 0 & c_5 \end{pmatrix} \\ &= -d^h l_2 l_3 s_3 s_5. \end{aligned} \tag{7.24} $$

Note that the determinant of the Jacobian is independent of the reference frame with respect to which it is calculated:

$$ {}_f\boldsymbol{J} = {}^{i}_{f}\boldsymbol{S}\ {}_i\boldsymbol{J} \Rightarrow \det({}_f\boldsymbol{J}) = \det({}^{i}_{f}\boldsymbol{S})\ \det({}_i\boldsymbol{J}), \tag{7.25} $$

and $\det(\boldsymbol{S}) = \det^2(\boldsymbol{R}) = 1$ (Sect. 6.6.3).

7.9 Inverse position kinematics

The inverse position kinematics ("IPK") solves the following problem: *Given the actual end-effector pose* ${}^{ee}_{bs}\boldsymbol{T}$, *what are the corresponding joint positions* $\boldsymbol{q} = (q_1 \ \ldots \ q_n)^T$*?* In contrast to the forward problem, the solution of the inverse problem is *not* always unique: the same end-effector pose can be reached in several *configurations*, corresponding to distinct joint position vectors. A 6R manipulator (a serial chain with six revolute joints, as in Figs 7.1, 7.2, and 7.3), with a *completely general* geometric structure has *sixteen* different inverse kinematics solutions, [36, 55], found as the solutions of a sixteenth order polynomial.

As for the forward position and velocity kinematics, this Section presents both a numerical procedure for general serial structures, and the dedicated closed-form solution for robots of the 321 type, as described in [21]. Some older references describe similar solution approaches but in less detail, [28, 54, 59].

The IK of a serial arm are more complex than its FK. However, many industrial applications don't need IK algorithms, since the desired positions and orientations of their end-effectors are *manually taught*: a human operator steers the robot to its desired pose, by means of control signals to each individual actuator; the operator stores the sequence of corresponding *joint* positions into the robot's memory; during subsequent task execution, the robot controller moves the robot to this set of taught joint coordinates. Note that the current trends towards off-line programming does require IK algorithms. *And* hence calibrated robots. Recall that such calibrated robots have a *general* kinematic structure.

7.9.1 General IPK: Newton-Raphson iteration

Inverse position kinematics for serial robot arms with a *completely general* kinematic structure (*but* with *six* joints) are solved by iterative procedures, based on the Newton-Raphson approach, [54, 66]:

Numerical IPK

Step 1 Start with an *estimate* $\hat{\boldsymbol{q}} = (\hat{q}_1 \ldots \hat{q}_6)^T$ of the vector of six joint positions. This estimate is, for example, the solution corresponding to a previous nearby pose, or, for calibrated robots, the solution calculated by the nominal model (using the procedure of the next Section if this nominal model has a 321 structure). As with all iterative algorithms, the better the initial guess, the faster the convergence.

Step 2 Denote the end-effector pose that corresponds to this estimated vector of joint positions by $\boldsymbol{T}(\hat{\boldsymbol{q}})$. The difference between the desired end-effector pose $\boldsymbol{T}(\boldsymbol{q})$ (with $\boldsymbol{q}$ the real joint positions which have to be found) and the estimated pose is "infinitesimal," as assumed in any iterative procedure:

$$\boldsymbol{T}(\boldsymbol{q}) = \boldsymbol{T}(\hat{\boldsymbol{q}}) \ \boldsymbol{T}_{\triangle}(\Delta\boldsymbol{q}). \tag{7.26}$$

$\Delta\boldsymbol{q} \triangleq \boldsymbol{q} - \hat{\boldsymbol{q}}$ is the joint position increment to be solved by the iteration. Solving for $\boldsymbol{T}_{\triangle}(\Delta\boldsymbol{q})$ yields

$$\boldsymbol{T}_{\triangle}(\Delta\boldsymbol{q}) = \boldsymbol{T}^{-1}(\hat{\boldsymbol{q}})\boldsymbol{T}(\boldsymbol{q}). \tag{7.27}$$

Step 3 Equation (6.18) gives the form of the infinitesimal pose $\boldsymbol{T}_{\triangle}(\Delta\boldsymbol{q})$:

$$\boldsymbol{T}_{\triangle} = \begin{pmatrix} 1 & -\delta_z & \delta_y & d_x \\ \delta_z & 1 & -\delta_x & d_y \\ -\delta_y & \delta_x & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{7.28}$$

The infinitesimal displacement twist $\mathbf{t}_\Delta(\hat{\boldsymbol{q}}) = (\delta_x\, \delta_y\, \delta_z\, d_x\, d_y\, d_z)^T$ corresponding to $\boldsymbol{T}_\Delta(\Delta \boldsymbol{q})$ is easily identified from Eq. (7.28). On the other hand, it depends linearly on the joint increment $\Delta \boldsymbol{q}$ through the Jacobian matrix $\boldsymbol{J}(\hat{\boldsymbol{q}})$, Eq. (7.15):

$$\mathbf{t}_\Delta(\hat{\boldsymbol{q}}) = \boldsymbol{J}(\hat{\boldsymbol{q}})\Delta \boldsymbol{q} + \mathcal{O}(\Delta \boldsymbol{q}^2). \tag{7.29}$$

$\boldsymbol{J}(\hat{\boldsymbol{q}})$ is calculated by the numerical FVK algorithm in Sect. 7.8.2.

Step 4 Hence, the joint increment $\Delta \boldsymbol{q}$ is approximated by

$$\boxed{\Delta \boldsymbol{q} = \boldsymbol{J}^{-1}(\hat{\boldsymbol{q}})\, \mathbf{t}_\Delta(\hat{\boldsymbol{q}}).} \tag{7.30}$$

The inverse of the Jacobian matrix exists only when the robot arm has *six* independent joints. Section 7.14 explains how to cope with the case of more or less than six joints.

Step 5 If $\Delta \boldsymbol{q}$ is "small enough," the iteration stops, otherwise Steps 2–4 are repeated with the new estimate $\hat{\boldsymbol{q}}_{i+1} = \hat{\boldsymbol{q}}_i + \Delta \boldsymbol{q}$.

This procedure gives an idea of the approach, but real implementations must take care of several numerical details, such as, for example:

1. Inverting a 6×6 Jacobian matrix (which is required in motion control) is not an insurmountable task for modern microprocessors (even if the motion controller runs at a frequency of 1000Hz or more), but nevertheless the implementation should be done very carefully, in order not to loose numerical accuracy.
2. In order to solve a set of linear equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, it is, from a numerical point of view, not a good idea to first calculate the inverse $\boldsymbol{A}^{-1}$ of the matrix $\boldsymbol{A}$ explicitly, and then to solve the equation by multiplying the vector $\boldsymbol{b}$ by this inverse, as might be suggested by Eq. (7.30). Numerically more efficient and stable algorithms exist, [22, 66], the simplest being the *Gaussian elimination* technique and its extensions.
3. The numerical procedure finds only *one* solution, i.e., the one to which the iteration converges. Some more elaborate numerical techniques exist to find *all* solutions, such as for example the *continuation*, *dialytic elimination* and *homotopy methods*, [56, 67, 70].

7.9.2 Closed-form IPK for 321 structure

The efficient closed-form IPK solution for the 321 structure relies again on the decoupling at the wrist centre point, [21]:

Closed-form IPK

Step 1 The position of the wrist centre point is simply given by the inverse of Eq. (7.12):

$$_{bs}\boldsymbol{p}^{wr} = {}_{bs}\boldsymbol{p}^{ee} - {}^{ee}_{bs}\boldsymbol{R}\; (0\,0\,l_6)^T. \tag{7.31}$$

Step 2 Hence, the first joint angle is

$$q_1 = \operatorname{atan2}({}_{bs}\boldsymbol{p}^{wr}_x, \pm{}_{bs}\boldsymbol{p}^{wr}_y). \tag{7.32}$$

The robot configuration corresponding to a positive ${}_{bs}\boldsymbol{p}^{wr}_y$ is called the "*forward*" solution, since the wrist centre point is then in front of the "body" of the robot; if ${}_{bs}\boldsymbol{p}^{wr}_y$ is negative, the configuration is called "*backward.*"

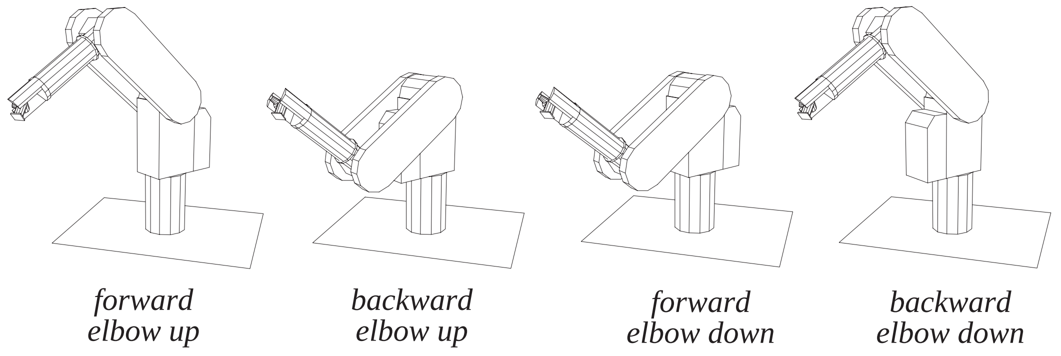


Figure 7.11: Four of the eight configurations corresponding to the same end effector pose, for a 321 type of manipulator. The four other configurations are similar to these four, except for a change in the wrist configuration from “flip” to “no flip.”

Step 3 The horizontal and vertical distances d^h and d^v of the wrist centre point with respect to the shoulder frame $\{1\}$ are found by inspection of Fig. 7.10:

$$d^h = \sqrt{({}_{bs}\boldsymbol{p}^{wr}_x)^2 + ({}_{bs}\boldsymbol{p}^{wr}_y)^2}, \quad d^v = {}_{bs}\boldsymbol{p}^{wr}_z - l_1. \tag{7.33}$$

Step 4 Now, look at the planar triangles formed by the second and third links (Fig. 7.10).

Step 4.1 The cosine rule gives

$$q_3 = \pm \arccos\left(\frac{(d^h)^2 + (d^v)^2 - (l_2)^2 - (l_3)^2}{2l_2l_3}\right). \tag{7.34}$$

A positive q_3 gives the “*elbow up*” configuration (Fig. 7.11); the configuration with negative q_3 is called “*elbow down*.”

Step 4.2 The tangent rules yield

$$q_2 = \text{atan2}\left(d^h, d^v\right) - \alpha, \tag{7.35}$$

with

$$\alpha = \text{atan2}\left(l_3s_3, l_2 + l_3c_3\right). \tag{7.36}$$

Step 5 The inverse position for the *ZXZ* wrist has already been described in Section 5.3.2. It needs ${}^6_4\boldsymbol{R}$ as input, which is straightforwardly derived from ${}^7_{bs}\boldsymbol{R}$ (a known input parameter) and ${}^4_{bs}\boldsymbol{R}$ (which is known if the first three joint angles are known):

$${}^6_4\boldsymbol{R} = {}^7_4\boldsymbol{R} = {}^{bs}_4\boldsymbol{R}\ {}^7_{bs}\boldsymbol{R}, \tag{7.37}$$

with

$${}^4_{bs}\boldsymbol{R} = \boldsymbol{R}(Z, q_1)\boldsymbol{R}(X, -q_2 - q_3). \tag{7.38}$$

As mentioned in Section 5.2.8, two solutions exist: one with $q_5 > 0$ (called the “*no-flip*” configuration) and one with $q_5 < 0$ (called the “*flip*” configuration).

In the algorithm above, a "*configuration*" (Sect. 7.3) corresponds to a particular choice of IPK solution. In total, the 321 manipulator has eight different configurations, by combining the binary decisions "forward/backward," "elbow up/elbow down," and "flip/no flip." Note that these names are not standardised: the robotics literature contains many alternatives.

7.10 Inverse velocity kinematics

Assuming that the inverse *position* kinematics problem has been solved for the current end-effector pose ${}^{ee}_{bs}\boldsymbol{T}$, the inverse *velocity* kinematics ("IVK") then solves the following problem: *Given the end-effector twist* $\mathbf{t}^{ee}$*, what is the corresponding vector of joint velocities* $\dot{\boldsymbol{q}} = (\dot{q}_1 \; \dots \; \dot{q}_n)^T$*?* A very common alternative name for the IVK algorithm is "*resolved rate*" procedure, especially in the context of robot control, [73].

As in the previous Sections, a numerical procedure for general serial structures is given, as well as a dedicated closed-form solution for robots of the 321 type. Note that the problem is only well-defined if the robot has *six* joints: if $n < 6$ not all end-effector twists can be generated by the robot; if $n > 6$ all end-effector twists can be generated in infinitely many ways.

7.10.1 General IVK: numerical inverse Jacobian

As for the inverse position kinematics, the inverse velocity kinematics for general kinematic structures must be solved in a numerical way. The simplest procedure corresponds to one iteration step of the numerical procedure used for the inverse position kinematics problem:

Numerical IVK

Step 1 Calculate the Jacobian matrix $\boldsymbol{J}(\boldsymbol{q})$.

Step 2 Calculate its inverse $\boldsymbol{J}^{-1}(\boldsymbol{q})$ numerically.

Step 3 The joint velocities $\dot{\boldsymbol{q}}$ corresponding to the end-effector twist $\mathbf{t}^{ee}$ are:

$$\boxed{\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\boldsymbol{q})\,\mathbf{t}^{ee}.} \tag{7.39}$$

Note that, as mentioned before, better and more efficient algorithms calculate $\dot{\boldsymbol{q}}$ without the explicit calculation of the matrix inverse $\boldsymbol{J}^{-1}$.

7.10.2 Closed-form IVK for 321 structure

The symbolically derived Jacobian for the 321 kinematic structure (Sect. 7.8.3) turns out to be easily invertible symbolically too when expressed in the wrist centre frame, [29, 57, 58]:

Step 1 The Jacobian ${}_4\boldsymbol{J}$ (Eqs (7.21) and (7.22)) has a zero 3×3 block in the lower right-hand side:

$${}_4\boldsymbol{J} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{0}_3 \end{pmatrix}. \tag{7.40}$$

Step 2 It is then easily checked by straightforward calculation that

$${}_4\boldsymbol{J}^{-1} = \begin{pmatrix} \boldsymbol{0}_3 & \boldsymbol{C}^{-1} \\ \boldsymbol{B}^{-1} & -\boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{C}^{-1} \end{pmatrix}. \tag{7.41}$$

Step 3 The inverses $\boldsymbol{B}^{-1}$ and $\boldsymbol{C}^{-1}$ are found symbolically by dividing the transpose of their matrices of cofactors by their determinants. These determinants are readily obtained from Eqs (7.21)–(7.22):

$$\det(\boldsymbol{B}) = s_5, \qquad \det(\boldsymbol{C}) = l_2 l_3 d^h s_3. \tag{7.42}$$

Hence,

$$\boldsymbol{B}^{-1} = \frac{1}{s_5}\begin{pmatrix} s_4 c_5 & s_5 c_4 & -s_4 \\ -c_5 c_4 & s_5 s_4 & c_4 \\ s_5 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{C}^{-1} = \begin{pmatrix} -\dfrac{1}{d^h} & 0 & 0 \\ 0 & 0 & \dfrac{1}{l_2 s_3} \\ 0 & -\dfrac{1}{l_3} & -\dfrac{l_2 c_3 + l_3}{l_2 l_3 s_3} \end{pmatrix}. \tag{7.43}$$

Step 4 In order to find the joint velocities, one has to *post*-multiply ${}_4\boldsymbol{J}^{-1}$ by ${}^{bs}_{4}\boldsymbol{S}$:

$$\dot{\boldsymbol{q}} = {}_4\boldsymbol{J}^{-1}\ {}^{bs}_{4}\boldsymbol{S}\ {}_{bs}\mathbf{t} = {}_{bs}\boldsymbol{J}^{-1}\ {}_{bs}\mathbf{t}. \tag{7.44}$$

7.11 Inverse force kinematics

Assuming that the inverse *position* kinematics problem has been solved for the current end-effector pose ${}^{ee}_{bs}\boldsymbol{T}$, the inverse *force* kinematics ("IFK") then solves the following problem: *Given the wrench* $\mathbf{w}^{ee}$ *that acts on the end-effector, what is the corresponding vector of joint forces/torques* $\boldsymbol{\tau} = (\tau_1 \ \dots \ \tau_n)^T$*?* This Section presents two equivalent approaches.

Projection on joint axes. The end-effector twist $\mathbf{t}^{ee}$ is the *sum* of the twists generated by all joints individually; but a wrench $\mathbf{w}$ exerted on the end-effector is *transmitted* unchanged to each joint in the serial chain. Part of the transmitted wrench is to be taken up *actively* by the joint actuator, the rest is taken up *passively* by the mechanical structure of the joint. While the wrench is physically the same screw at each joint, its *coordinates* expressed in the local joint frames differ from frame to frame. The wrench coordinates ${}_{bs}\mathbf{w}$ of the end-effector wrench expressed in the base reference frame $\{bs\}$ are related to the coordinates ${}_i\mathbf{w}$ of the wrench expressed in the reference frame of the ith joint through the screw transformation matrix ${}^{bs}_{i}\boldsymbol{S}$, Eq. (6.25):

$${}_i\mathbf{w} = {}^{bs}_{i}\boldsymbol{S}\ {}_{bs}\mathbf{w}. \tag{7.45}$$

If this local frame $\{i\}$ has its Z^i axis along the prismatic or revolute joint axis, then the force component τ_i felt by the joint actuator corresponds to, respectively, the third and sixth coordinate of ${}_i\mathbf{w}$. These coordinates are found by premultiplying ${}_{bs}\mathbf{w}$ by the third or sixth *rows* of ${}^{bs}_{i}\boldsymbol{S}$, or equivalently, the third and sixth *columns* of ${}^{bs}_{i}\boldsymbol{S}^T$. Equations (6.26)–(6.27) learn that these are given by

$${}^{bs}_{i}\boldsymbol{S}_{3\times} = {}^{bs}_{i}\boldsymbol{S}^T_{\times 3} = \begin{pmatrix} {}_{bs}\boldsymbol{e}^i_z \\ \boldsymbol{0} \end{pmatrix}, \quad \text{and} \quad {}^{bs}_{i}\boldsymbol{S}_{6\times} = {}^{bs}_{i}\boldsymbol{S}^T_{\times 6} = \begin{pmatrix} {}_{bs}\boldsymbol{p}^{bs,i} \times {}_{bs}\boldsymbol{e}^i_z \\ {}_{bs}\boldsymbol{e}^i_z \end{pmatrix}, \tag{7.46}$$

where $\boldsymbol{S}_{3\times}$ indicates the third *row* of matrix $\boldsymbol{S}$, and $\boldsymbol{S}_{\times 6}$ the sixth column. These columns resemble the columns $\boldsymbol{J}_i$ of the Jacobian matrix as used for *screw twists*, but with the first and second three-vectors interchanged. So, premultiplication of $\boldsymbol{J}_i$ by $\widetilde{\boldsymbol{\Delta}}$, Eq. (4.18), makes the resemblance exact. The above reasoning can be repeated for all joints, and for all other twist and wrench representations. Hence, the IFK is

$$\boxed{\boldsymbol{\tau} = (\widetilde{\boldsymbol{\Delta}}\,\boldsymbol{J})^T\mathbf{w}.} \tag{7.47}$$

Conservation of virtual work. A second approach to derive this IFK is through the instantaneous power generated by an end-effector twist $\mathbf{t}$ against the wrench $\mathbf{w}$ exerted on the end-effector. This power equals $\mathbf{t}^T \widetilde{\boldsymbol{\Delta}}\, \mathbf{w}$ in Cartesian space coordinates (Sect. 4.5.4), and $\sum_{i=1}^{n} \tau_i \dot{q}_i$ in joint space coordinates. Replacing $\mathbf{t}$ by $\boldsymbol{J}\dot{\boldsymbol{q}}$ and some simple algebraic manipulations yield Eq. (7.47) again. In the robotics literature you see this relationship most often in the form $\boldsymbol{\tau} = \boldsymbol{J}^T \mathbf{w}$, due to the difference in twist representation from the one used in this text (Sect. 6.5.1): $\boldsymbol{J}^{\text{literature}} = \widetilde{\boldsymbol{\Delta}}\, \boldsymbol{J}^{\text{this text}}$.

Fact-to-Remember 51 ("Jacobian transpose")
Equation (7.47) is often referred to as the "Jacobian transpose" relationship between end-effector wrench $\mathbf{w}$ *and joint force/torque vector* $\boldsymbol{\tau}$*. It represents the fact that the joint torque that keeps a static wrench exerted on the end-effector in equilibrium is given by the projection of this end-effector wrench on the joint axis. This fact is valid for any serial robot arm.*

Strictly speaking, $\widetilde{\boldsymbol{\Delta}}\boldsymbol{J}$ is not a "Jacobian matrix," since wrenches are not the partial derivatives of anything.

7.12 Forward force kinematics

The forward force kinematics ("FFK") solves the following problem: *Given the vectors of joint force/torques* $\boldsymbol{\tau} = (\tau_1 \ \dots \ \tau_n)^T$*, what is the resulting static wrench* $\mathbf{w}^{ee}$ *that the end-effector exerts on the environment?* (If the end-effector is rigidly fixed to a rigid environment!) This problem is only well-defined if the robot has *six* joints: if $n < 6$ the robot cannot generate a full six-dimensional space of end-effector wrenches; if $n > 6$ all wrenches can be generated in infinitely many ways.

7.12.1 Dual wrench

For a robot with six joints, the Jacobian matrix is a *basis* for the twist space ("tangent space") of the end-effector (Sect. 7.8.1). Section 3.9 introduced the concept of a *dual basis* of the wrench space ("co-tangent space"), when a basis for the twist space is given. The kinematic structure of the robot is a *de facto* choice of twist space basis. The natural pairing ("reciprocity") between twists and wrenches leads to a de facto dual basis in the wrench space:

Fact-to-Remember 52 (Physical interpretation of dual wrench basis)
The ith column of the "dual" wrench basis of a serial robot arm is the wrench on the end-effector that generates a unit force/torque at the ith joint, and zero forces/torques at the other joints.
Each column of the dual wrench basis is sometimes called a partial wrench, [45].

This text uses the notation $\boldsymbol{G} = (\boldsymbol{G}_1 \ \dots \ \boldsymbol{G}_6)$ for the matrix of the six dual basis wrenches $\boldsymbol{G}_i$. Its definition yields the following relationship with the Jacobian matrix $\boldsymbol{J}$ of the same robot arm:

$$\boldsymbol{J}\, \widetilde{\boldsymbol{\Delta}}\, \boldsymbol{G} = \mathbf{1}_{6\times 6}. \tag{7.48}$$

7.12.2 General FFK: dual wrenches

$\boldsymbol{G}$ is a basis for the wrench space, hence each wrench $\mathbf{w}$ on the end-effector has coordinates $\boldsymbol{\tau} = (\tau_1, \dots, \tau_6)$:

$$\mathbf{w} = \boldsymbol{G}\,\boldsymbol{\tau}. \tag{7.49}$$

τ_i is the force/torque required at the ith joint to keep the end-effector wrench $\mathbf{w}$ in static equilibrium (neglecting gravity of the links!). The relation with the "Jacobian transpose" formula for the Inverse Force Kinematics, Eq. (7.47) is also immediately clear:

$$\boldsymbol{G} = (\widetilde{\boldsymbol{\Delta}}\boldsymbol{J})^{-T}. \tag{7.50}$$

7.12.3 Closed-form FFK for 321 structure

As before, using the wrist centre point of the 321 kinematic structure allows for a solution of the FFK problem by simple inspection, since the partial wrench of each joint is easily found from Fig. 7.10 in Sect. 7.6.2:

Joint 1 The partial wrench is a *pure force* through the wrist centre point and parallel to the axes of the second and third joints.

Joint 2 The partial wrench is a *pure force* through the wrist centre point and through the joint axis of the first and third joints.

Joint 3 The partial wrench is a *pure force* through the wrist centre point and through the joint axis of the first and second joints.

Joints 4,5,6 The partial wrench of each of these joints is the combination of:

1. A *pure moment* about a line through the wrist centre point and orthogonal to the axes of the two other wrist joints. This moment has no components about these other two joint axes. However, it can have components about the first three joint axes.
2. These components about the first three joint axes are compensated by *pure forces* that do not generate moments about these first three joint axes. These forces are: (i) through the origins of the second and third joint frames (i.e., along l_2), and (ii) through the first joint axis and parallel with the second and third joint axes.

7.13 Singularities

The inverse velocity kinematics exhibit singularities:

Fact-to-Remember 53 (Singularity: physical interpretation)
At a <u>singularity</u>, the Jacobian matrix $\boldsymbol{J}$ <u>looses</u> rank.

This means that the end-effector looses one or more degrees of *twist* freedom (i.e., instantaneously, the end-effector cannot move in these directions). Equivalently, the space of wrenches on the end-effector that are taken up *passively* by the mechanical structure of the robot (i.e., without needing any joint torques to be kept in static equilibrium) increases its dimension.

Serial robots with less than six independent joints are always “singular” in the sense that they can never span a six-dimensional twist space. This is often called an “architectural singularity” [41].

A singularity is usually not an isolated point in the workspace of the robot, but a sub-manifold.

As for the inverse position kinematics, the *general* approach to find the singularities of a serial manipulator is *numerical.* For 321 robots, however, a *closed-form* solution follows straightforwardly from the closed-form velocity kinematics. The following Sections give the details.

7.13.1 Numerical singularity detection

For a *square* Jacobian, $\det(\boldsymbol{J}) = 0$ is a necessary and sufficient condition for a singularity to appear. However, some robots do not have square Jacobians. Hence, a better numerical criterion is required. The most popular criterion is based on the *Singular Value Decomposition* (SVD), [22, 42, 66], that works for all possible kinematic structures: *every* matrix $\boldsymbol{A}$ (with arbitrary dimensions $m \times n$) has an SVD decomposition of the form

$$\underset{m\times n}{\boldsymbol{A}} = \underset{m\times m}{\boldsymbol{U}} \; \underset{m\times n}{\boldsymbol{\Sigma}} \; \underset{n\times n}{\boldsymbol{V}^T}, \quad \text{with} \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \sigma_m & 0 & \dots & 0 \end{pmatrix}, \tag{7.51}$$

represented here for $n > m$. $\boldsymbol{U}$ and $\boldsymbol{V}$ are *orthogonal matrices*, and the singular values σ_i are in descending order: $\sigma_1 \geqslant \sigma_2 \geqslant \dots \geqslant \sigma_m \geqslant 0$. $\boldsymbol{A}$ has full rank (i.e., $\text{rank}(\boldsymbol{A}) = m$ in the case above where $n > m$) if $\sigma_m \neq 0$; it loses rank if $\sigma_m \approx 0$, i.e., σ_m is zero within a numerical tolerance factor. Hence, the most popular way to monitor a robot’s “closeness” to singularity is to check the smallest singular value in the SVD of its Jacobian matrix. Note that this requires quite some computational overhead, and that better methods exist for closed-form kinematics designs, such as the 321 structure.

7.13.2 Singularity detection for 321 structure

The singular positions of the 321 robot structure, Fig. 7.9, follow immediately from the closed-form inverse velocity kinematics: the determinant of the Jacobian is $-d^h l_2 l_3 s_3 s_5$, Eq. (7.24). Hence, it vanishes in the following three cases (Fig. 7.12):

Arm-extended singularity ($q_3 = 0$) ([38] calls it a “regional” singularity.) The robot reaches the end of its regional workspace, i.e., the positions that the wrist centre point can reach by moving the first three joints. The screw reciprocal to the remaining five motion degrees of freedom is a force along the arm.

Wrist-extended singularity ($q_5 = 0$) ([38] calls it a “boundary” singularity.) The first and last joint of the wrist are aligned, so they span the same motion freedom. Hence, the angular velocity about the common normal of the three wrist joints is lost. The screw reciprocal to the remaining five motion degrees of freedom cannot be described in general: it depends not only on the wrist joints, but on the first three joint angles too, [38].

Wrist-above-shoulder singularity ($d^h = 0$) ([38] calls it an “orientation” singularity.) The first joint axis intersects the wrist centre point. This means that the three wrist joints (which are equivalent to a spherical joint) and the first joint are not independent. The screw reciprocal to the five remaining motion degrees of freedom is a force through the wrist centre point, and orthogonal to the plane formed by the first three links.

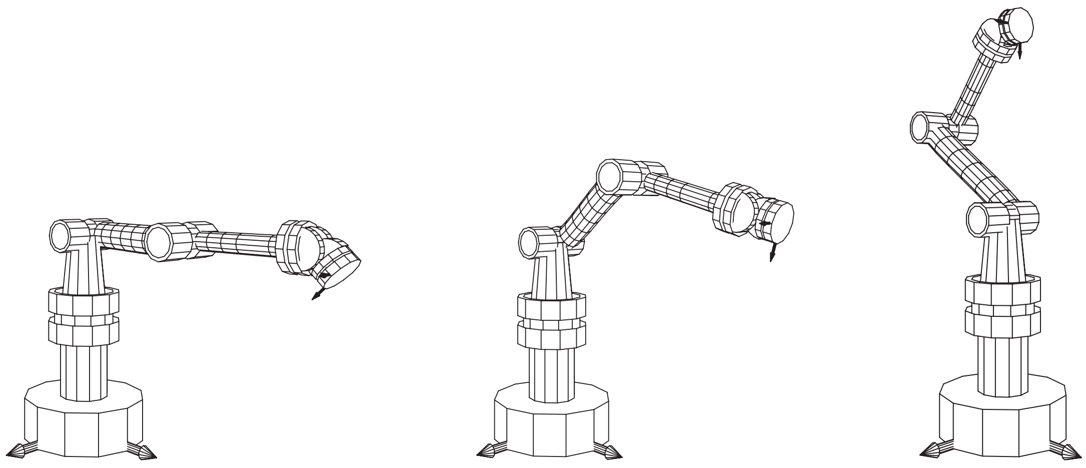


Figure 7.12: The three singular positions for the 6R wrist-partitioned serial robot arm with closed form kinematic solutions: "arm-extended," "wrist-extended" and "wrist-above-shoulder."

Fact-to-Remember 54 (Singularity and configuration)
Contrary to what might be suggested by the previous paragraphs, it is not necessary that a robot passes through a singularity in order to change configuration, [20, 30, 72]. Only special structures, such as the 321 robots, have their singularities coinciding with their configuration borders.

Look at the Jacobian matrix for a 321 design, Eq. (7.40). The zero block in this Jacobian comes from the fact that the wrist is spherical, i.e., it generates no translational components when expressed in the wrist centre frame. A spherical wrist does not only decouple the position and orientation kinematics, but also the singularities:

Fact-to-Remember 55 (Singularity decoupling)
A spherical wrist $\underline{\text{decouples}}$ the singularities of the wrist, $\det(\boldsymbol{B}) = 0$, and the singularities of the regional structure of the arm, $\det(\boldsymbol{C}) = 0$, [75].

A general kinematic structure has more complicated and less intuitive singularities. The reason why *shoulder offsets* (Sect.7.6, Fig. 7.2) have been introduced as extensions to the 321 kinematic structure is that they make sure that the robot cannot reach "wrist-above-shoulder" singular position. The reason behind *elbow offsets* is to avoid the "arm-extended" singularity in the "zero position" of the robot; zero positions (of part of the arm) are often used as reference positions at start-up of the robot, and it is obviously not a good idea to let the robot start in a singularity.

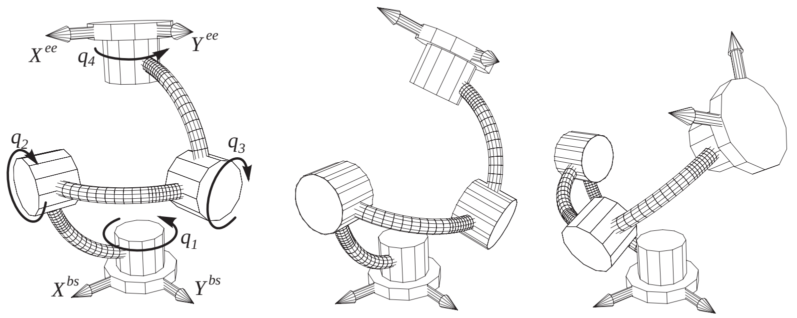


Figure 7.13: Redundant wrist with four revolute joints. In the left-most configuration, two axes line up, but the wrist does not become singular.

7.14 Redundancies

Definition 2 (Redundant robot arm) *A manipulator with n joints is called redundant if it is used to perform a task that requires less than the available n degrees of freedom.*

For example, a classical six degrees of freedom serial robot is redundant if it has to follow the surface of a workpiece with a vertex-like tool (Fig. 7.15) and no orientation constraints are specified: only three joints are required to keep the tool vertex on the surface, and only five are needed to keep the tool aligned with, for example, the normal direction to the surface. In general, however, robots are designed for more than just one single task, and hence we speak of redundant robots when they have seven or more joints. An obvious choice for an *anthropomorphic* redundant robot is the "7R" manipulator, Fig. 7.14. It has an extra joint between the "shoulder" and the "elbow" of the 6R wrist-partitioned manipulators in Figures 7.1 and 7.2. In this way, the robot can reach "around" obstacles that the 6R robots cannot avoid. Such a redundant manipulator can attain any given pose in its dextrous workspace in infinitely many ways. This is obvious from the following argument: if one fixes one particular joint to an *arbitrary* joint value, a full six degrees of freedom robot still remains, and this robot can reach the given pose in at least one way. This 7R manipulator can also avoid the "extended-wrist" and "wrist-above-shoulder" singularities (but not necessarily both at the same time).

7.14.1 Forward kinematics

The forward kinematics (both position and velocity) give no special difficulties: any given set of joint positions and joint velocities still corresponds to one unique end-effector pose and twist, and the forward velocity kinematics are still described by the Jacobian matrix:

$$\underset{6\times 1}{\mathbf{t}^{ee}} = \underset{6\times 7}{\boldsymbol{J}(\boldsymbol{q})} \; \underset{7\times 1}{\dot{\boldsymbol{q}}}. \tag{7.52}$$

This Jacobian matrix has more columns than rows, e.g., it is a 6×7 matrix in the case of the 7R robot. Hence, it always has a *null space*, i.e., a set of joint velocities that do not move the end-effector:

$$\boxed{\mathrm{Null}\,(\boldsymbol{J}(\boldsymbol{q})) = \left\{\dot{\boldsymbol{q}}^N \mid \boldsymbol{J}(\boldsymbol{q})\,\dot{\boldsymbol{q}}^N = \mathbf{0}\right\}.} \tag{7.53}$$

This null space depends on the current joint positions. Equation (7.53) implies that an arbitrary vector of the null space of the Jacobian can be used as an *internal motion* of the robot:

$$\mathbf{t}^{ee} = \boldsymbol{J}(\boldsymbol{q})\,\dot{\boldsymbol{q}} = \boldsymbol{J}(\boldsymbol{q})\left(\dot{\boldsymbol{q}} + \dot{\boldsymbol{q}}^N\right). \tag{7.54}$$

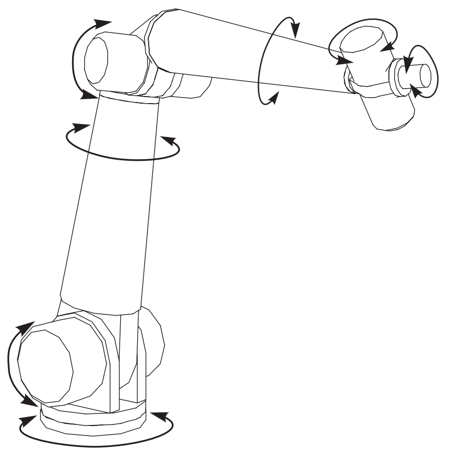


Figure 7.14: Redundant serial arm with seven revolute joints. It differs from the 321 structure in having an extra *shoulder* joint.

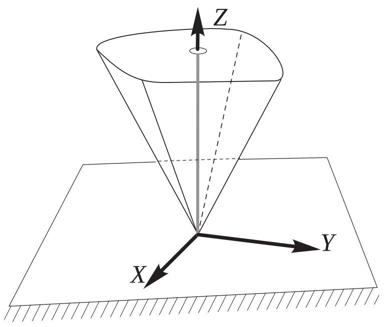


Figure 7.15: *Vertex-face* contact.

7.14.2 Inverse kinematics

The inverse kinematics of a redundant robot require the user to specify a criterion with which to solve the ambiguities in the joint positions and velocities (internal motions) corresponding to the specified end-effector pose and twist. Some examples of redundancy resolution criterions are:

1. Keep the joints as close as possible to a specified position. The goal of this criterion is to avoid that joints reach their *mechanical limits.* A simple approach to reach this goal is to attach *virtual springs* to the joints, with the equilibrium position of the springs near the middle of the motion range of the joints. With this spring model, the redundancy resolution criterion corresponds to the minimization of the potential energy in the springs.

2. Minimize the *kinetic energy* of the manipulator, [31].

3. Maximize the *manipulability* of the manipulator, i.e., keep the robot close to the joint positions that give it the best ability to move and/or exert forces in all directions, [19, 34, 48, 53].

4. Minimize the *joint torques* required for the motion. The goal of this criterion is to avoid saturation of the actuators, and to execute the task with minimum "effort," [16, 27].

5. Execute a high priority task but use the redundancy to achieve a lower priority task in parallel, [49].

6. Avoid obstacles in the robot's workspace. For example, a robot with an extra shoulder or elbow joint can reach "around" obstacles, [2, 26].

7. Avoid singularities in the robot kinematics, [1, 4, 37, 50, 63]. For example, the 4R "Hamilton wrist," [39], (Fig. 7.13) avoids the "extended-wrist" singularity.

8. Travel through singularities while keeping the joint velocities bounded, [7, 33].

Many of these redundancy resolution criterions (implicitly or explicitly) rely on the concept of the *extended Jacobian*, [1]. This approach starts from the observation that the $6 \times n$ Jacobian can be made into a $n \times n$ matrix by adding $n-6$ rows to it, collected here in a $(n-6)\times n$ matrix $\boldsymbol{A}$:

$$\bar{\boldsymbol{J}} = \left(\begin{array}{c} \boldsymbol{J}(\boldsymbol{q}) \\ \hline \boldsymbol{A}(\boldsymbol{q}) \end{array} \right). \tag{7.55}$$

This is equivalent to adding $n-6$ *linear constraints* on the joint velocities:

$$\boldsymbol{A}(\boldsymbol{q})\dot{\boldsymbol{q}} = 0. \tag{7.56}$$

In order to obtain a full-rank extended Jacobian $\bar{\boldsymbol{J}}$, the constraint matrix $\boldsymbol{A}$ must be full rank, and *transversal* (or "*transient*") to the Jacobian $\boldsymbol{J}$, i.e., the null spaces of $\boldsymbol{A}$ and $\boldsymbol{J}$ should have no elements in common, [64]. Equation (7.55) then has a uniquely defined inverse:

$$\bar{\boldsymbol{J}}^{-1} \triangleq \left(\boldsymbol{B} \mid *\right). \tag{7.57}$$

The $n\times 6$ matrix $\boldsymbol{B}$ is a so-called *generalized inverse*, or *pseudo-inverse*, often denoted by $\boldsymbol{B} = \boldsymbol{J}^{\dagger}$, [5, 9, 44]: it satisfies $\boldsymbol{JB} = \mathbf{1}_{6\times 6}$ and $\boldsymbol{BJ} = \mathbf{1}_{n\times n}$. (This follows straightforwardly from the definition of $\bar{\boldsymbol{J}}$.) With it, the forward velocity kinematics, Eq. (7.52), can be "inverted":

$$\dot{\boldsymbol{q}} = \boldsymbol{B}\,\mathbf{t}^{ee}. \tag{7.58}$$

Do not forget that the resulting joint velocities depend on the choice of the constraint matrix $\boldsymbol{A}$. The following paragraphs derive this general result of Eq. (7.58) in more detail and in an alternative way for the particular example of the kinetic energy minimization criterion. As will be described in Chapter 10, the kinetic energy T of a serial manipulator is of the form

$$T = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\,\dot{\boldsymbol{q}}. \tag{7.59}$$

Since T is a *positive scalar* (and hence $T^T = T$), the inertia matrix $\boldsymbol{M}$ is both invertible and symmetric. Minimizing the kinetic energy, while at the same time obeying the inverse kinematics requirement that $\mathbf{t}^{ee} = \boldsymbol{J}\,\dot{\boldsymbol{q}}$, transforms the solution to the following *constrained optimization problem*:

$$\left\{ \begin{array}{rcl} \min\limits_{\dot{\boldsymbol{q}}} \;\; T & = & \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\,\dot{\boldsymbol{q}}, \\ \text{such that} \;\; \mathbf{t}^{ee} & = & \boldsymbol{J}(\boldsymbol{q})\,\dot{\boldsymbol{q}}. \end{array} \right. \tag{7.60}$$

The classical solution of this kind of problem uses *Lagrange multipliers*, [12, 66], i.e., the constraint in (7.60) is integrated into the functional T to be minimized as follows:

$$\min_{\dot{\boldsymbol{q}}} \;\; T' = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{M}\,\dot{\boldsymbol{q}} + \boldsymbol{\lambda}^T\left(\mathbf{t}^{ee} - \boldsymbol{J}\,\dot{\boldsymbol{q}}\right). \tag{7.61}$$

(For notational simplicity, we dropped the dependence of $\boldsymbol{M}$ and $\boldsymbol{J}$ on the joint positions $\boldsymbol{q}$.) $\boldsymbol{\lambda}$ is the column vector of the (currently unknown) Lagrange multipliers. They can be physically interpreted as the *impulses* (forces times mass) generated by violating the constraint $\mathbf{t}^{ee} - \boldsymbol{J}\,\dot{\boldsymbol{q}} = 0$. (Check the physical units!) The Lagrange multipliers are determined together with the desired joint velocities by setting to zero the partial derivatives of the functional T' with respect to the minimization parameter vector $\dot{\boldsymbol{q}}$:

$$\underset{1\times 7}{\dot{\boldsymbol{q}}^{\,T}}\;\underset{7\times 7}{\boldsymbol{M}} - \underset{1\times 6}{\boldsymbol{\lambda}^T}\;\underset{6\times 7}{\boldsymbol{J}} = \boldsymbol{0}_{1\times 7}. \tag{7.62}$$

This gives a set of seven equations, in the seven joint velocities and the six Lagrange multipliers. These Lagrange multipliers can be solved for by post-multiplying Eq. (7.62) by $\boldsymbol{M}^{-1}\boldsymbol{J}^T$:

$$\dot{\boldsymbol{q}}^T \boldsymbol{J}^T = \boldsymbol{\lambda}^T \left(\boldsymbol{J}\,\boldsymbol{M}^{-1}\boldsymbol{J}^T \right). \tag{7.63}$$

The left-hand side of this equation equals the transpose of the end effector twist, $(\mathbf{t}^{ee})^T$, and the matrix triplet on the right-hand side is a square 6×6 matrix that can be inverted (at least if the manipulator is not in a singular configuration, Sect. 7.13). Hence,

$$\boldsymbol{\lambda}^T = (\mathbf{t}^{ee})^T \left(\boldsymbol{J}\,\boldsymbol{M}^{-1}\boldsymbol{J}^T \right)^{-1}. \tag{7.64}$$

Equations (7.62) and (7.64), and the fact that $\boldsymbol{M}$ is symmetric, yield

$$\dot{\boldsymbol{q}} = \boldsymbol{M}^{-1}\boldsymbol{J}^T \left(\boldsymbol{J}\,\boldsymbol{M}^{-1}\boldsymbol{J}^T \right)^{-1} \mathbf{t}^{ee} \tag{7.65}$$

$$\triangleq \boldsymbol{J}^{\dagger}_{M^{-1}} \mathbf{t}^{ee}. \tag{7.66}$$

$\boldsymbol{J}^{\dagger}_{M^{-1}}$ is a $n \times 6 (n > 6)$ matrix, the so-called *weighted pseudo-inverse* of $\boldsymbol{J}$, with $\boldsymbol{M}^{-1}$ acting as weighting matrix *on the space of joint velocities*, [5, 9, 44]. It is not a good idea to calculate the solution $\dot{\boldsymbol{q}}$ by the straightforward matrix multiplications of Eq. (7.65); better numerical techniques exist, see e.g. [22].

Fact-to-Remember 56 (Redundancy resolution)
The redundancy resolution approaches based on an extended Jacobian yield only local optimality. For example, one minimizes the instantaneous kinetic energy, not the kinetic energy over a complete motion. The success of the extended Jacobian approach is due to the fact that analytical solutions exist for quadratic cost functions only.

Cyclicity—Holonomic constraints. When one steers the end-effector of a redundant robot along a *cyclic motion* (i.e., it travels through the some trajectory of *end-effector* poses repetitively), the pseudo-inverse derived from an extended Jacobian typically results in different *joint* trajectories during each cycle, [3, 10, 14, 47, 64]. Whether or not the joint space trajectory is cyclic depends on the *integrability* of the constraint equation (7.56). If this equation is integrable, the constraints are called *holonomic.* This name comes from the Greek word *holos*, which means "whole, integer." A (non)holonomic constraint on the joint *velocities* can (not) be integrated to give a constraint on the joint *positions.* Section 5.3.6 already explained how integrability can be checked.

7.15 Constrained manipulator

The previous Section looked at the case in which one imposes *virtual* constraints on the manipulator; this Section explains how to deal with *physical* constraints: free-space motion with less than six joints, or motion in contact with a stiff environment and with a six-jointed manipulator. The former gives rise to an optimization problem in Cartesian space; the latter to an optimization problem in joint space, dual to the redundancy resolution in the previous Section.

7.15.1 Free-space motion with less than six degrees of freedom

Assume the manipulator has less than six joints, say $6-n$. Hence, the Jacobian $\boldsymbol{J}$ is a $6 \times n$ matrix, and there always exists a reciprocal wrench space of at least dimension n. Such a manipulator is constrained to move on a $(6-n)$-dimensional sub-manifold of SE(3). That means that it can not generate any arbitrary end-effector twist $\mathbf{t}^{ee}$. A kinematic energy based pseudo-inverse procedure exists to *project* $\mathbf{t}^{ee}$ on the span of $\boldsymbol{J}$. This procedure is derived quite similarly to the redundancy resolution procedure of the previous Section; nevertheless, it has fundamentally different properties. The (unconstrained) objective kinetic energy function to be minimized is:

$$\min_{\dot{\boldsymbol{q}}} \quad T = \frac{1}{2}(\mathbf{t}^{ee} - \boldsymbol{J}\dot{\boldsymbol{q}})^T \boldsymbol{M}\, (\mathbf{t}^{ee} - \boldsymbol{J}\dot{\boldsymbol{q}}). \tag{7.67}$$

$\boldsymbol{M}$ is a (full-rank) *Cartesian space* mass matrix, Chapt. 10. The physical interpretation of this minimization problem is that the end-effector twist $\mathbf{t}^{ee}$ is approximated by that twist $\boldsymbol{J}\dot{\boldsymbol{q}}$ on the constrained sub-manifold that results in the smallest "loss" of kinetic energy compared to the case that the full $\mathbf{t}^{ee}$ could be executed. Setting the partial derivative of the objective function with respect to the joint velocities to zero yields:

$$\boldsymbol{M}\mathbf{t}^{ee} = \boldsymbol{M}\boldsymbol{J}\dot{\boldsymbol{q}}.$$

(Recall that $\boldsymbol{M}$ is symmetric, hence $\boldsymbol{M}^T = \boldsymbol{M}$.) Pre-multiplying with $\boldsymbol{M}^{-1}$ is not allowed, since $\boldsymbol{J}$ is not of full column rank. However, pre-multiplying with $\boldsymbol{J}^T$ gives

$$\boldsymbol{J}^T\boldsymbol{M}\mathbf{t}^{ee} = (\boldsymbol{J}^T\boldsymbol{M}\boldsymbol{J})\dot{\boldsymbol{q}}.$$

The matrix $(\boldsymbol{J}^T\boldsymbol{M}\boldsymbol{J})$ is square $(n \times n)$ and full-rank if the manipulator is not in a singular configuration. Hence, it is invertible, and

$$\begin{aligned} \dot{\boldsymbol{q}} &= (\boldsymbol{J}^T\boldsymbol{M}\boldsymbol{J})^{-1}\boldsymbol{J}^T\boldsymbol{M}\mathbf{t}^{ee}, \\ &\triangleq \boldsymbol{J}^{\dagger}_{M}\mathbf{t}^{ee}. \end{aligned} \tag{7.68}$$

$\boldsymbol{J}^{\dagger}_{M}$ is also a *weighted pseudo-inverse* of $\boldsymbol{J}$, but this time with $\boldsymbol{M}$ as the 6×6 weighting matrix *on the space of Cartesian twists*. Pre-multiplying Eq. (7.68) with $\boldsymbol{J}$ proves that the executed twist $\mathbf{t}$ is a projection of the desired twist $\mathbf{t}^{ee}$:

$$\mathbf{t} = \boldsymbol{J}\dot{\boldsymbol{q}} = \boldsymbol{J}\boldsymbol{J}^{\dagger}_{M}\mathbf{t}^{ee} \triangleq \boldsymbol{P}\mathbf{t}^{ee}. \tag{7.69}$$

It is straightforward to check that $\boldsymbol{P}$ indeed satisfies the projection operator property that $\boldsymbol{P}\boldsymbol{P} = \boldsymbol{P}$. A set of linear constraints similar to Eq. (7.56) does not exist: all joint velocities are possible.

7.15.2 Motion in contact

Assume the manipulator has six joints, but its end-effector makes contact with a (stiff) environment. This means that it looses a number of degrees of motion freedom, say n. The Jacobian $\boldsymbol{J}$ is still a 6×6 matrix, but an n-dimensional wrench space exists (with wrench basis $\boldsymbol{G}$) to which the *allowed* motions of the end-effector must be reciprocal. This imposes a set of linear constraints on the joint velocities as in Eq. (7.56):

$$(\boldsymbol{G}^T\widetilde{\boldsymbol{\Delta}}\boldsymbol{J})\dot{\boldsymbol{q}} = 0. \tag{7.70}$$

A possible way to "filter" any possible end-effector twist $\mathbf{t}^{ee} \in \text{span}(\boldsymbol{J})$ into a twist $\mathbf{t}$ compatible with the constraint (i.e., reciprocal to $\boldsymbol{G}$) follows a procedure similar to the redundancy resolution in Sect. 7.14.2. Indeed, this "*kinetostatic filtering*" [18] can be formulated as the following constrained optimization problem:

$$\left\{ \begin{array}{rcl} \min\limits_{\mathbf{t}} \; T & = & \frac{1}{2}(\mathbf{t}^{ee} - \mathbf{t})^T \boldsymbol{M} \; (\mathbf{t}^{ee} - \mathbf{t}) \\ \text{such that} \quad \boldsymbol{G}^T \widetilde{\boldsymbol{\Delta}} \mathbf{t} & = & 0. \end{array} \right. \tag{7.71}$$

The solution of this optimization problem runs along similar lines as the redundancy resolution problems in Sect. 7.14: (i) include the constraint in the objective function by means of Lagrange multipliers; (ii) set the partial derivative with respect to **t** equal to zero; and (iii) solve for the Lagrange multipliers. This leads to the following weighted projection operation:

$$\mathbf{t} = \left(\mathbf{1} - \left((\widetilde{\boldsymbol{\Delta}} \boldsymbol{G})^{\boldsymbol{T}} \right)^{\dagger}_{M^{-T}} (\widetilde{\boldsymbol{\Delta}} \boldsymbol{G})^T \right) \mathbf{t}^{ee}. \tag{7.72}$$

Definition 3 (Workspace constraints) *A robot is constrained in its motion at a given configuration if it cannot generate velocities that span the complete tangent space at that configuration.*

We distinguish between

1. *Kinematic constraints* (also called *geometric* constraints): the instantaneous velocities that the robot can execute form (part of) a vector space with dimension lower than six. This space is called the *twist space* of the constraint. Equivalently, there exists a non-empty *wrench space* of generalized forces at the end-effector that are balanced passively by the mechanical structure of the robot. "Passive" means: without requiring torques at the driven joints. The twist and wrench spaces are always *reciprocal*, Sect. 4.5.4. An element of the wrench space is said to be a *reciprocal wrench.*

2. *Dynamic constraints*: the actuators cannot produce sufficient torque to generate any possible velocity, or rather acceleration. This means that the *bandwidth* of the robot motion is limited, but not necessarily the spatial directions in which it can move.

A mechanical limit of a revolute joint is a simple example of a kinematic constraint: when the joint has reached this mechanical limit, the end-effector can resist any wrench that corresponds to a pure torque about this joint (and in the direction of the mechanical limit!). Another simple example of a kinematically constrained robot is a robot with less than six joints, e.g., the SCARA robot of Fig. 7.4. The twist space of this robot is never more than four-dimensional: it can always resist pure moments about the end-effector's X and Y axes (if Z is the direction of the translational and angular motion of the last link).

A kinematic motion constraint is correctly modelled by (i) a basis for the twist space of instantaneous velocities allowed by the constraint, or (ii) a basis of the wrench space of instantaneous forces the constraint can absorb. A kinematic constraint is *not* correctly modelled by a so-called "space of impossible motions" (i.e, motions that the robot cannot execute) or a "space of non-reciprocal screws" (i.e., wrenches that are not reciprocal to the constraint twist space): neither of these concepts is well-defined, since (i) the sum of an impossible motion with *any* possible motion remains an impossible motion, and (ii) adding any reciprocal screw to a non-reciprocal screw gives another non-reciprocal screw.

The discussion in this Section assumes that all bodies and all kinematic constraints are infinitely stiff. If this is not the case, the robot can move against such a compliant environment, and the relationships between the possible motions and the corresponding forces are determined by the *contact impedance.*

References for this Chapter

[1] J. Baillieul. Kinematic programming alternatives for redundant manipulators. In *IEEE Int. Conf. Robotics and Automation*, pages 722–728, St. Louis, MS, 1985.

[2] J. Baillieul. Avoiding obstacles and resolving kinematic redundancy. In *IEEE Int. Conf. Robotics and Automation*, pages 1698–1704, San Fransisco, CA, 1986.

[3] D. R. Baker. Some topological problems in robotics. *The Mathematical Intelligencer*, 12(1):66–76, 1990.

[4] D. R. Baker and C. W. Wampler II. On the inverse kinematics of redundant manipulators. *Int. J. Robotics Research*, 7(2):3–21, 1988.

[5] A. Ben-Israel and T. N. E. Greville. *Generalized Inverses: Theory and Applications.* Robert E. Krieger Publishing Company, Huntington, NY, reprinted edition, 1980.

[6] R. Bernhardt and S. L. Albright. *Robot Calibration.* Chapman & Hall, 1993.

[7] N. Boland and R. Owens. On the behaviour of robots through singularities. In R. A. Jarvis, editor, *Int. Symp. Industrial Robots*, pages 1122–1134, Sydney, Australia, 1988.

[8] R. C. Buck and E. F. Buck. *Advanced calculus.* International series in pure and applied mathematics. McGraw-Hill, New York, NY, 3rd edition, 1978.

[9] S. L. Campbell and C. D. Meyer, Jr. *Generalized Inverses of Linear Transformations.* Dover, 1991.

[10] Y. S. Chung, M. Griffis, and J. Duffy. Repeatable joint displacement generation for redundant robotic systems. *Trans. ASME J. Mech. Design*, 116:11–16, 1994.

[11] P. M. Cohn. *Algebra.* Wiley, Chichester, 2nd edition, 1991.

[12] R. Courant and D. Hilbert. *Methods of mathematical physics.* Interscience, New York, NY, 1970.

[13] J. J. Craig. *Introduction to Robotics: Mechanics and Control.* Addison-Wesley, Reading, MA, 1986.

[14] A. De Luca and G. Oriolo. Nonholonomic behavior in redundant robots under kinematic control. *IEEE Trans. Rob. Automation*, 13(5):776–782, 1997.

[15] J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Trans. ASME J. Appl. Mech.*, 23:215–221, 1955.

[16] A. S. Deo and I. D. Walker. Minimum effort inverse kinematics for redundant manipulators. *IEEE Trans. Rob. Automation*, 13(5):767–775, 1997.

[17] K. L. Doty. Tabulation of the symbolic midframe Jacobian of a robot manipulator. *Int. J. Robotics Research*, 6(4):85–97, 1987.

[18] K. L. Doty, C. Melchiorri, and C. Bonivento. A theory of generalized inverses applied to robotics. *Int. J. Robotics Research*, 12(1):1–19, 1993.

[19] K. L. Doty, C. Melchiorri, E. M. Schwartz, and C. Bonivento. Robot manipulability. *IEEE Trans. Rob. Automation*, 11(3):462–468, 1995.

[20] J. El Omri and P. Wenger. A general criterion for the identification of nonsingular posture changing 3-DOF manipulators. In J.-P. Merlet and B. Ravani, editors, *Computational Kinematics '95*, pages 153–162, Sophia-Antipolis, France, 1995. Kluwer Academic Publishers, Dordrecht.

[21] R. Featherstone. Position and velocity transformations between robot end-effector coordinates and joint angles. *Int. J. Robotics Research*, 2(2):35–45, 1983.

[22] G. Golub and C. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, 1989.

[23] Y.-L. Gu and J.-S. Ho. A unified representation and its applications to robotic control. In *IEEE Int. Conf. Robotics and Automation*, pages 98–103, Cincinnati, OH, 1990.

[24] R. S. Hartenberg and J. Denavit. *Kinematic synthesis of linkages.* McGraw-Hill, New York, NY, 1964.

[25] S. Hayati, K. Tso, and G. Roston. Robot geometry calibration. In *IEEE Int. Conf. Robotics and Automation*, pages 947–951, Philadelphia, PA, 1988.

[26] J. M. Hollerbach. Optimum kinematic design for a seven degree of freedom manipulator. In Hanafusa and Inoue, editors, *Robotics Research: The Second International Symposium*, pages 215–222. MIT Press, Cambridge, MA, 1985.

[27] J. M. Hollerbach and K. C. Suh. Redundancy resolution of manipulators through torque optimization. In *IEEE Int. Conf. Robotics and Automation*, pages 1016–1021, St. Louis, MS, 1985.

[28] B. K. P. Horn and H. Inoue. Kinematics of the MIT-AI-Vicarm manipulator. Technical Report Working Paper 69, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1974.

[29] K. H. Hunt. Robot kinematics—A compact analytic inverse solution for velocities. *Trans. ASME J. Mech. Transm. Automation Design*, 109:42–49, 1987.

[30] C. Innocenti and V. Parenti-Castelli. Singularity-free evolution from one configuration to another in serial and fully-parallel manipuators. *Trans. ASME J. Mech. Design*, 120:73–79, 1998.

[31] O. Khatib. *Commande dynamique dans l'espace opérationnel des robots manipulateurs en présence d'obstacles.* PhD thesis, Ecole Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, 1980.

[32] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE J. Rob. Automation*, RA-3(1):43–53, 1987.

[33] J. Kieffer. Differential analysis of bifurcations and isolated singularities for robots and mechanisms. *IEEE Trans. Rob. Automation*, 10(1):1–10, 1994.

[34] C. Klein and B. Blaho. Dexterity measures for the design and control of kinematically redundant manipulators. *Int. J. Robotics Research*, 6(2):72–83, 1987.

[35] V. Kumar and K. J. Waldron. The workspace of a mechanical manipulator. *Trans. ASME J. Mech. Design*, 103:665–672, 1981.

[36] H. Y. Lee and C. G. Liang. A new vector theory for the analysis of spatial mechanisms. *Mechanism and Machine Theory*, 23(3):209–217, 1988.

[37] J. Lloyd. *Robot Trajectory Generation for Paths with Kinematic Singularities.* PhD thesis, McGill University, Montreal, Canada, 1995.

[38] G. L. Long and R. P. Paul. Singularity avoidance and the control of an eight-revolute-joint manipulator. *Int. J. Robotics Research*, 11(6):503–515, 1992.

[39] G. L. Long, R. P. Paul, and W. D. Fisher. The Hamilton wrist: a four-revolute-joint spherical wrist without singularities. In *IEEE Int. Conf. Robotics and Automation*, pages 902–907, Scottsdale, AZ, 1989.

[40] K. H. Low and R. N. Dubey. A comparative study of generalized coordinates for solving the inverse-kinematics problem of a 6R robot manipulator. *Int. J. Robotics Research*, 5(4):69–88, 1986.

[41] O. Ma and J. Angeles. Optimum architecture design of platform manipulators. In *Int. Conf. Advanced Robotics*, pages 1130–1135, Pisa, Italy, 1991.

[42] A. A. Maciejewski and C. A. Klein. The singular value decomposition: Computation and applications to robotics. *Int. J. Robotics Research*, 8(6):63–79, 1989.

[43] S. MacLane and G. Birkhoff. *Algebra.* MacMillan, New York, NY, 2nd edition, 1979.

[44] M. L. Moe. Kinematics and rate control of the Rancho arm. In *Proceedings of the First CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators*, pages 253–272, Wien, Austria, 1973. Springer Verlag.

[45] M. G. Mohamed and J. Duffy. A direct determination of the instantaneous kinematics of fully parallel robot manipulators. *Trans. ASME J. Mech. Transm. Automation Design*, 107:226–229, 1985.

[46] B. W. Mooring, Z. S. Roth, and M. R. Driels. *Fundamentals of Manipulator Calibration.* John Wiley & Sons, Inc., New York, NY, 1991.

[47] F. A. Mussa-Ivaldi and N. Hogan. Integrable solutions of kinematic redundancy via impedance control. *Int. J. Robotics Research*, 10(5):481–491, 1991.

[48] Y. Nakamura. *Advanced robotics: redundancy and optimization.* Addison-Wesley, Reading, MA, 1991.

[49] Y. Nakamura, H. Hanafusa, and T. Yoshikawa. Task-priority based redundancy control of robot manipulators. *Int. J. Robotics Research*, 6(2):3–15, 1987.

[50] K. A. O'Neil, Y.-C. Chen, and J. Seng. Removing singularities of resolved motion rate control of mechanisms, including self-motion. *IEEE Trans. Rob. Automation*, 13(5):741–751, 1997.

[51] D. E. Orin and W. W. Schrader. Efficient computation of the Jacobian for robot manipulators. *Int. J. Robotics Research*, 3(4):66–75, 1984.

[52] F. C. Park. Optimal robot design and differential geometry. *Trans. ASME J. Mech. Design*, 117:87–92, 1995.

[53] F. C. Park and J. W. Kim. Kinematic manipulability of closed chains. In J. Lenarčič and V. Parenti-Castelli, editors, *Recent Advances in Robot Kinematics*, pages 99–108, Portorož-Bernardin, Slovenia, 1996. Kluwer Academic Publishers.

[54] D. L. Pieper. *The kinematics of manipulators under computer control.* PhD thesis, Stanford University, Department of Mechanical Engineering, 1968.

[55] M. Raghavan and B. Roth. Inverse kinematics of the general 6R manipulator and related linkages. *Trans. ASME J. Mech. Design*, 115:502–508, 1993.

[56] M. Raghavan and B. Roth. Solving polynomial systems for the kinematic analysis and synthesis of mechanisms and robot manipulators. *Trans. ASME J. Mech. Design*, 117:71–79, 1995.

[57] M. Renaud. *Contribution à l'étude de la modélisation et à la commande dynamiques des robots manipulateurs.* PhD thesis, Université Paul Sabatier, Toulouse, 1980.

[58] M. Renaud. Geometric and kinematic models of a robot manipulator: calculation of the Jacobian matrix and its inverse. In *Int. Symp. Industrial Robots*, pages 453–465, Tokyo, Japan, 1981.

[59] B. Roth, J. Rastegar, and V. Scheinman. On the design of computer controlled manipulators. In *Proceedings of he First CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators*, pages 94–112, Wien, Austria, 1973. Springer Verlag.

[60] W. Rudin. *Principles of mathematical analysis.* International series in pure and applied mathematics. McGraw-Hill Kogakusha, Tokyo, Japan, 3rd edition, 1976.

[61] K. Schröer, S. L. Albright, and M. Grethlein. Complete, minimal and model-continuous kinematic models for robot calibration. *Rob. Comp.-Integr. Manufact.*, 13(1):73–85, 1997.

[62] L. Sciavicco and B. Siciliano. *Modeling and Control of Robot Manipulators.* McGraw-Hill, 1996.

[63] T. Shamir. The singularities of redundant robot arms. *Int. J. Robotics Research*, 9(1):113–121, 1990.

[64] T. Shamir and Y. Yomdin. Repeatability of redundant manipulators: Mathematical solution of the problem. *IEEE Trans. Autom. Control*, 33(11):1004–1009, 1988.

[65] M. W. Spong and M. Vidyasagar. *Robot dynamics and control.* Wiley, New York, NY, 1989.

[66] G. Strang. *Introduction to applied mathematics.* Wellesley-Cambridge Press, Wellesley, MA, 1986.

[67] L.-W. Tsai and A. P. Morgan. Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods. *Trans. ASME J. Mech. Transm. Automation Design*, 107:189–195, 1985.

[68] W. K. Veitschegger and C.-H. Wu. Robot accuracy analysis based on kinematics. *IEEE J. Rob. Automation*, RA-2(3):171–179, 1986.

[69] K. J. Waldron, S.-L. Wang, and S. J. Bolin. A study of the Jacobian matrix of serial manipulators. *Trans. ASME J. Mech. Transm. Automation Design*, 107:230–238, 1985.

[70] C. W. Wampler, A. P. Morgan, and A. J. Sommese. Numerical continuation methods for solving polynomial systems arising in kinematics. *Trans. ASME J. Mech. Design*, 112:59–68, 1990.

[71] H. J. Warnecke and R. D. Schraft. *Industrie-Roboter.* Krausskopf Verlag, 1973.

[72] P. Wenger. A new general formalism for the kinematic analysis of all nonredundant manipulators. In *IEEE Int. Conf. Robotics and Automation*, pages 442–447, Nice, France, 1992.

[73] D. E. Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Trans. Man-Machine Systems*, 10(2):47–53, 1969.

[74] D. E. Whitney. The mathematics of coordinated control of prosthetic arms and manipulators. *Trans. ASME J. Dyn. Systems Meas. Control*, 94:303–309, 1972.

[75] R. L. Williams, II. The double universal joint wrist on a manipulator: Inverse position kinematics and singularities. In *ASME Design Technical Conferences, 23rd Biennial Mechanisms Conference, DE-Vol. 72*, pages 355–360, Minneapolis, MN, 1994.

[76] W. A. Wolovich. *Robotics: basic analysis and design.* Holt, Rinehart and Winston, New York, NY, 1986.

Chapter 8

Parallel manipulator kinematics

8.1 Introduction

The previous Chapter discussed the kinematics of *serial* robot arms, i.e., the base and end-effector are connected by one single serial chain of actuated joints. This Chapter introduces the kinematics of *parallel* robot arms, i.e., the base and end-effector are connected by multiple serial chains, in which not all joints are actuated. A *fully parallel* robot has six serial chains in parallel, and only one joint in each chain is actuated (Fig. 8.1). Of course, all sorts of combinations of these purely serial and parallel structures are possible, and many exist in practice.

The main reasons for the overwhelming success of the serial robot design (over 99% of installed industrial robots...) is that (i) it gives a large workspace compared to the space occupied by the robot itself, and (ii) kinematic designs exist that simplify the mathematics of the robot's geometry enormously. The main drawback of a serial design is its low intrinsic rigidity, so that heavy links and joints must be used to obtain a reasonable

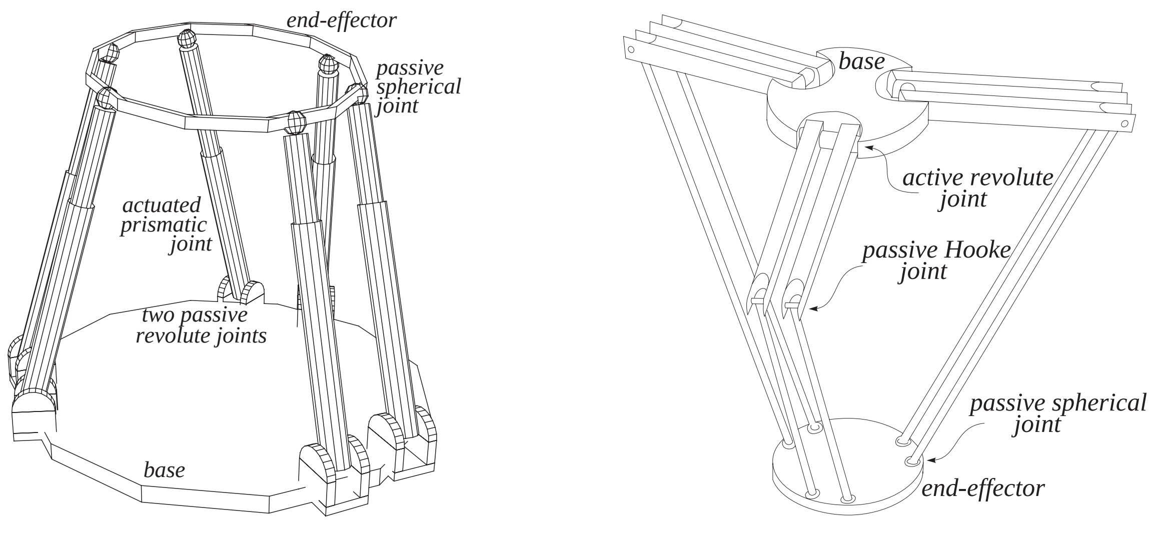


Figure 8.1: Fully parallel robots. Left: General *Stewart-Gough platform*; the actuated joints are prismatic, the passive joints are revolute. Right: *HEXA platform*; all joints are revolute.

effective rigidity at the end point. These pros and cons are exactly the opposites of those of parallel manipulators. The fully parallel designs of Fig. 8.1 have all actuators in or near the base, which results in a very low inertia of the part of the robot that has actually to be moved. Hence, a higher bandwidth can be achieved with the same actuation power. This is why parallel structures are used for flight simulators.

A parallel structure supports its end-effector in multiple places, which yields a stiffer and hence more accurate manipulator for the same weight and cost, *and* which causes the positioning errors generated in each leg to "average out," again increasing the accuracy. This would be very advantageous for accurate milling (Fig. 8.2). However, experiments with real prototypes show that parallel structures currently do not live up to these expectations: their accuracy and stiffness are about an order of magnitude worse than for classical serial machines. The reasons are: (i) the compliance of the ball screws in the prismatic joints, (ii) the complexity of the construction with many passive joints that all have to be manufactured and assembled with strict tolerances, and (iii) the high forces that some passive joints have to resist.

The main disadvantage of parallel manipulators is their small workspace: legs can collide, and there are many passive joints in the structure that all introduce joint limit constraints. This is especially the case with the spherical "ball-in-socket" joints used in most implementations, [37].

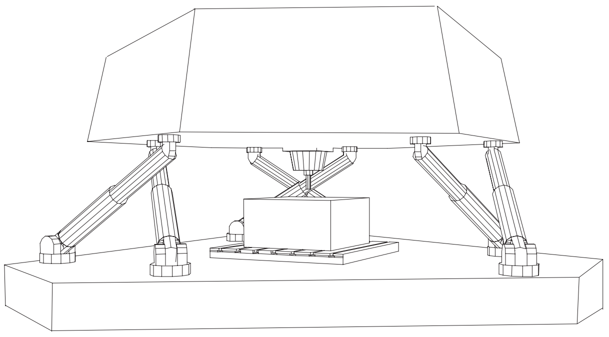


Figure 8.2: Milling machine with a parallel manipulator design (*Variax*, by Gidding & Lewis).

Duality. Parallel and serial robots are *dual*, not only in the sense that the weak points of serial designs are the strong points of parallel designs and vice versa, but also from the geometrical and mathematical point of view, which is based on the dualities between twists and wrenches, Sect. 3.9. This Chapter exploits these dualities by describing the kinematics of parallel robots by the same geometrical concepts used in the serial manipulator case.

Fact-to-Remember 57 (Basic ideas of this Chapter)
The dualities between serial and parallel manipulators imply that no new concepts or mathematics at all have to be introduced in order to understand and describe parallel manipulators. Roughly speaking, one just has to interchange the words "twist" and "wrench," "forward" and "inverse," "straightforward" and "complicated," "large" and "small," etc.

Kinematics. The definitions of forward and inverse position and velocity kinematics as defined for serial robots apply to parallel robots without change. But parallel robots have a large number of *passive joints*, whose only function is to allow the required number of degrees of freedom to each leg. Adding a leg between end-effector and base *adds* motion constraints to the end-effector, while in the case of serial robots adding a joint *reduces* the motion constraints (or, equivalently, *adds* a motion degree of freedom). This text discusses six degrees of freedom robots only, but many designs have less than six, e.g., planar or spherical robots, [3, 12].

8.2 Parallel robot designs

In its most general form, a parallel robot design consists of a number of serial subchains, all connected to the same rigid end-effector. As in the case of serial robots, simplicity considerations have resulted in the use of only a limited set of designs.

The first design was completed towards the beginning of the 1950s, by Gough in the United Kingdom, and implemented and used as a tyre testing machine in 1955, [13]. In fact, it was a huge force sensor, capable of measuring forces and torques on a wheel in all directions. Some years later, Gough's compatriot Stewart published a design for a flight simulator, [35]. In comments to Stewart's paper, Gough and others described their designs, such as the tyre testing machine mentioned above. Gough's design was fully parallel (while Stewart's was not), of the type depicted in Fig. 8.1. Nevertheless, the name of Stewart is still connected to the concept of fully parallel robots. Probably the first application of a parallel kinematic structure as a robotic *manipulator* was by McCallion and Pham, [23], towards the end of the 1970s. A decade later, all-revolute joint parallel manipulators were designed. Figure 8.1 shows the six degrees of freedom *HEXA* design, [31]. This was the successor of the three degrees of freedom *DELTA* robot, [9]. This DELTA design is a special case of the HEXA: the links in each couple of neighbouring legs in the HEXA design are rigidly coupled. This gives a *spatial parallellogram* which makes the end-effector platform move in translation only.

8.2.1 Design characteristics

The examples above illustrate the most common design characteristics of parallel robots:

1. All designs use *planar* base and end-effector platforms, i.e., the joints connecting the legs to the base all lie in the same plane, and similarly at the side of the end-effector. There is no physical motivation for this planarity constraint, i.e., base and end-effector could in principle have any possible shape. But these planarity constraints reduce the complexity of the mathematical description. This is another example of the fact that each geometric constraint imposed on the kinematic structure can be used in the kinematic routines to simplify the calculations, cf. Sect. 7.2.

2. Although any serial kinematic structure could be used as leg structure of a parallel design, only those serial structures are used for which the *inverse kinematics* (position and velocity) are *very simple*.

3. The previous characteristics are the reasons for the abundant use of *spherical* and *universal* joints. These joints not only simplify the kinematics, but they also make sure that the legs in the Stewart-Gough platforms experience only compressive or tensile loads, but no shear forces or bending and torsion moments. This reduces the deformation of the platform, even under high loads.

This last fact is easily proven by calculating the partial wrench (Sect.7.12.1) of the actuated prismatic joint in a leg of a Stewart-Gough design, Fig. 8.1. If the leg has length l the Jacobian of the leg, expressed in a reference

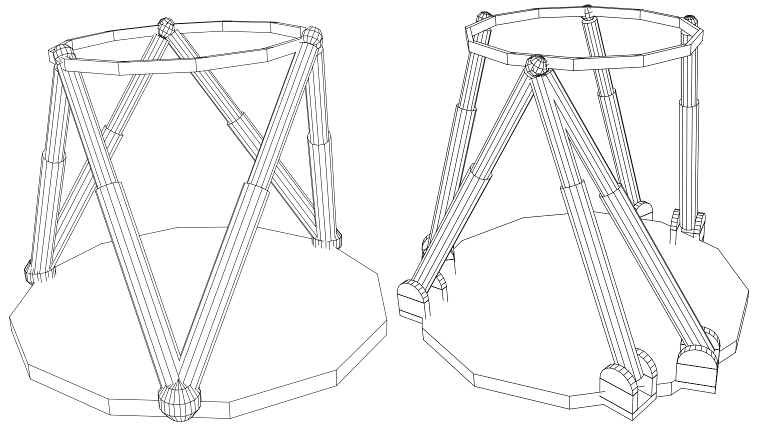


Figure 8.3: Left: 3-3 "Stewart(-Gough)" design. Right: 3-1-1-1 design.

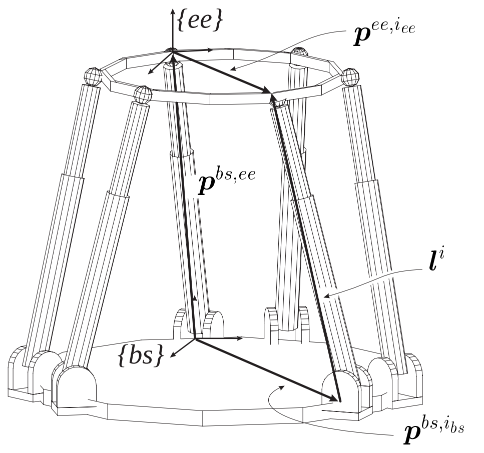


Figure 8.4: Generic kinematic model for a parallel manipulator.

frame in the spherical joint, is:

$$\boldsymbol{J} = \begin{pmatrix} 0 & l & 0 & 1 & 0 & 0 \\ -l & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The columns of $\boldsymbol{J}$ corresponds to the joints starting from the base: first the two intersecting revolute joints at a distance l from the reference frame, then the actuated prismatic joint acting along the Z-axis of this frame, and finally the three angular degrees of freedom with axes through the origin of the reference frame. The partial wrench of the third column is easily seen to be $\boldsymbol{G}_3 = (0\ 0\ 1\ 0\ 0\ 0)^T$, which is a pure force along the prismatic joint axis.

8.2.2 Nomenclature

One often subdivides the different designs according to the number of *coinciding* pivot points on the base and the end-effector. For example, the "Stewart platform" architecture as used in some flight simulators (leftmost drawing of Fig. 8.3) has pairwise coinciding connections at both the base and the end-effector, and is therefore called a 3-3 platform, or *octahedral* platform, [18, 22]. Manipulators with no coinciding connections are called 6-6 designs. The rightmost example in Figure 8.3 has three legs intersecting at the end-effector and is called a 3-1-1-1 design, [19].

8.3 Coordinate conventions and transformations

The link frame conventions and transformations defined for serial kinematic chains (Sect. 7.4) apply without change to each of the legs in a parallel robot. The only difference with the serial case are the definitions used for the connection of all legs to the base and the end-effector platforms. Figure 8.4 shows the kinematic model that this text will use as a generic example. These platforms are rigid bodies, which are represented by the reference frames $\{bs\}$ and $\{ee\}$, respectively. $\{bs\}$ serves as immobile world reference frame. The X and Z axes of $\{bs\}$ and $\{ee\}$ lie in the corresponding platform plane. The actuated joints in all six legs are prismatic. They are connected to the base and end-effector by a universal joint at the base and a spherical joint at the end-effector. The axis of the ith prismatic joint is a directed line $\mathcal{L}(\boldsymbol{p}^{bs,i_{bs}}, \boldsymbol{l}_i)$. $\boldsymbol{p}^{bs,i_{bs}}$ is the vector from the origin of $\{bs\}$ to the connection point of the ith leg with the base platform. $\boldsymbol{l}_i$ is a non-unit direction vector of the ith leg, and its length l^i equals the current length of the leg. $\boldsymbol{p}^{ee,i_{ee}}$ is the vector from the origin of $\{ee\}$ to the connection of the ith leg with the end-effector platform. Finally, the vector $\boldsymbol{p}^{bs,ee}$ connects the origin of $\{bs\}$ to the origin of $\{ee\}$. So, for each leg i, the following *position closure* constraint is always satisfied:

$$\boxed{\boldsymbol{p}^{bs,i_{bs}} + \boldsymbol{l}_i = \boldsymbol{p}^{bs,ee} + \boldsymbol{p}^{ee,i_{ee}}, \quad \forall i = 1, \ldots, 6.} \tag{8.1}$$

In this equation, $\boldsymbol{p}^{bs,i_{bs}}$ and $\boldsymbol{p}^{ee,i_{ee}}$ are known design constants, i.e., one knows their coordinates with respect to $\{bs\}$ and $\{ee\}$, respectively. $\boldsymbol{l}_i$ is time-varying and usually only its *magnitude* is measurable, not its direction. $\boldsymbol{p}^{bs,ee}$ changes with the position and orientation of the end-effector platform with respect to the base platform. The vector $\boldsymbol{q} = (q_1 \ldots q_6)^T$ denotes the joint positions (i.e., leg lengths, or joint angles of actuated revolute joints) of the parallel manipulator. Note that, in practice, these leg lengths are not necessarily equal to the positions measured by the linear encoders on the prismatic joints of the manipulator's legs, but the relationship between both is just a constant offset.

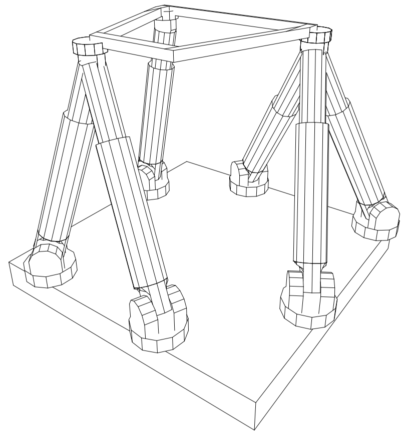


Figure 8.5: 321 parallel structure.

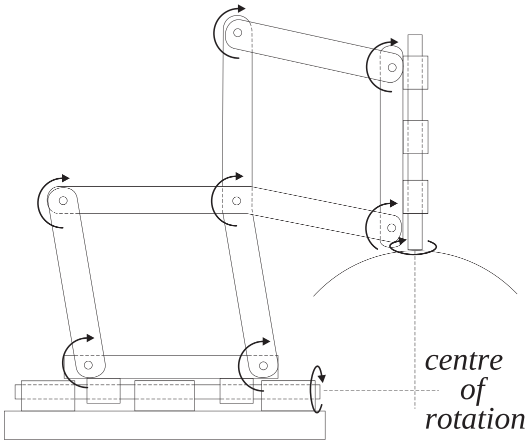


Figure 8.6: The "CMS" spherical joint, [15]. This design allows to connect several links to functionally concentric spherical joints.

8.4 321 kinematic structure

For serial robots, the 321 kinematic structure, Sect. 7.5, allows closed-form solutions for the inverse position and velocity kinematics. The parallel structure in Fig. 8.5 is the *dual* of this serial 321 structure: it has three intersecting prismatic legs (i.e., the dual of the three intersecting revolute joints of a spherical wrist), two intersecting prismatic legs (i.e., the dual of the two parallel (= intersecting at infinity) revolute joints in the regional structure of the serial 321 structure), and one solitary leg. Hence the name "321," [27, 28, 15]. Notwithstanding its extreme *kinematic* simplicity, the 321 manipulator is not yet used in real-world applications. The difficulties in *constructing* three concentric spherical joints is probably the major reason, with its moderate stiffness (with respect to the octahedral 3-3 design) an important secondary reason. Figure 8.6 shows a potential solution to the concentric joint problem, [15], and an alternative design is given in [5].

Fact-to-Remember 58 (Decoupled kinematics structure of parallel robots)
Similarly to the 321 design for serial robots, the 321 design for parallel robots allows for the decoupling of the position and orientation kinematics. The geometric feature that generates this decoupling is the tetrahedron formed by the three intersecting legs.

8.5 Inverse position kinematics

The inverse position kinematics problem ("IPK") can be stated in exactly the same way as for a serial manipulator: *Given the actual end-effector pose* ${}^{ee}_{bs}\boldsymbol{T}$*, what is the corresponding vector of leg positions* $\boldsymbol{q} = (q_1 \ \dots \ q_n)^T$*?* The IPK is a first example of the geometrical duality between serial and parallel manipulators: the *inverse* position kinematics for a parallel manipulator (with an arbitrary number of legs) has a unique solution (*if* each serial leg has a unique IPK!), and can be calculated straightforwardly; for a serial manipulator, these were properties of the *forward* position kinematics. The IPK works as follows:

Inverse position kinematics

Step 1 Equation (8.1) immediately yields the *vector* $\boldsymbol{l}_i$, since all other vectors in the position closure equation are known when ${}^{ee}_{bs}\boldsymbol{T}$ is known. In terms of coordinates with respect to the base reference frame $\{bs\}$, this equation gives

$$\begin{aligned} {}_{bs}\boldsymbol{l}_i &= {}_{bs}\boldsymbol{p}^{bs,ee} + {}_{bs}\boldsymbol{p}^{ee,i_{ee}} - {}_{bs}\boldsymbol{p}^{bs,i_{bs}} \\ &= {}_{bs}\boldsymbol{p}^{bs,ee} + {}^{ee}_{bs}\boldsymbol{R}_{ee}\boldsymbol{p}^{ee,i_{ee}} - {}_{bs}\boldsymbol{p}^{bs,i_{bs}}. \end{aligned} \tag{8.2}$$

${}_{bs}\boldsymbol{p}^{bs,ee}$ and ${}^{ee}_{bs}\boldsymbol{R}$ come from the input ${}^{ee}_{bs}\boldsymbol{T}$. ${}_{ee}\boldsymbol{p}^{ee,i_{ee}}$ and ${}_{bs}\boldsymbol{p}^{bs,i_{bs}}$ are known *constant* coordinate three-vectors, determined by the design of the manipulator.

Step 2 The *length* l_i of this vector $\boldsymbol{l}_i$ is the square root of the Euclidean norm:

$$l_i = \sqrt{(\boldsymbol{l}_{i,x})^2 + (\boldsymbol{l}_{i,y})^2 + (\boldsymbol{l}_{i,z})^2}. \tag{8.3}$$

In a Stewart-Gough design, this length immediately gives the desired position q_i of the actuated prismatic joint.

IPK of HEXA leg. In a HEXA design, l_i is the length between the end-points of a two-link manipulator in which both links are coupled by a two degrees of freedom universal joint, Fig 8.7. The relationship between the joint angle q_i and this length l_i follows from the following procedure. The joint angle q_i moves the end point of the first link (with length l^b_i, Fig 8.7) in leg i to the position $\boldsymbol{p}_i$ given by

$$\boldsymbol{p}_i = \boldsymbol{b}_i + {}^{i}_{bs}\boldsymbol{R}\ \boldsymbol{R}(X, q_i) \begin{pmatrix} 0 & 0 & l^b_i \end{pmatrix}^T. \tag{8.4}$$

${}^{i}_{bs}\boldsymbol{R}$ is the rotation matrix between the base frame $\{bs\}$ and a reference frame constructed in the actuated R joint, with X-axis along the joint axis and with the Z-axis the direction of the first link corresponding to a zero joint

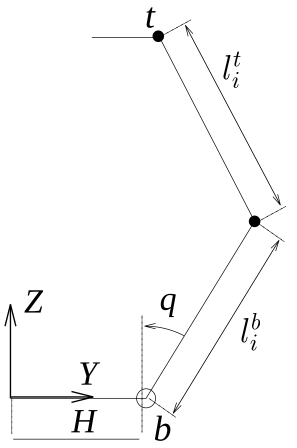


Figure 8.7: One leg in the HEXA design. The joint angle q_i is variable and measured; the lengths l_i^b and l_i^t of the "base" and "top" limbs of each leg are constant; the angles of all other joints are variable but not measured. Note that the joint between l_i^t and l_i^b is a two degrees of freedom universal joint, so that the link l_i^t does not necessarily lie in the plane of the figure.

angle q_i. This matrix ${}^{i}_{bs}\boldsymbol{R}$ is a constant of the mechanical design of the manipulator. $\boldsymbol{R}(X, q_i)$ is the rotation matrix corresponding to a rotation about the X axis over an angle q_i (Sect. 5.2.8):

$$\boldsymbol{R}(X, q_i) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(q_i) & -\sin(q_i) \\ 0 & \sin(q_i) & \cos(q_i) \end{pmatrix}. \tag{8.5}$$

In this equation, the joint angle q_i is the only unknown parameter. The positions $\boldsymbol{p}_i$ are connected to a top platform pivot point $\boldsymbol{t}_i$ by links of known lengths l_i^t. Hence

$$||\boldsymbol{p}_i - {}_{bs}\boldsymbol{t}_i|| = l_i^t, \tag{8.6}$$

with

$${}_{bs}\boldsymbol{t}_i = {}_{bs}\boldsymbol{p}^{bs,ee} + {}^{ee}_{bs}\boldsymbol{R}\ \boldsymbol{t}_i \tag{8.7}$$

the coordinates of the top platform pivot point $\boldsymbol{t}_i$ with respect to the base frame $\{bs\}$. Some straightforward rewriting of Eq. (8.6), including a substitution of $\sin(q_i) = 2t/(1+t^2)$ and $\cos(q_i) = (1-t^2)/(1+t^2)$, gives a *quartic* equation in $t = \tan(q_i/2)$. (Quartic equations can still be solved quite efficiently.) The two real solutions for q_i correspond to the two intersections of (i) the circle generated by rotating a link of length l_i^b about the axis of the actuated joint, and (ii) the sphere of radius l_i^t around $\boldsymbol{t}_i$. The other two solutions of the quartic are *always* complex.

8.6 Forward force kinematics

For serial manipulators, the end-effector *twist* is the sum of the contributions of each joint *velocity.* Dually, for parallel manipulators, the end-effector *wrench* is the sum of the contributions of each actuated joint's torque or force. If each leg of the parallel manipulators has six joints, this contribution of each actuated joint is exactly the partial wrench of the joint in its own leg, i.e., the force felt at the end-effector when all other joints generate no force. Hence, the formula for the FFK of a parallel manipulator (with $n \geq 6$ six-jointed legs) follows immediately:

$$\boxed{\mathbf{w}^{ee} = (\boldsymbol{G}_1 \ \dots \ \boldsymbol{G}_n) \begin{pmatrix} \tau_1 \\ \vdots \\ \tau_n \end{pmatrix} = \boldsymbol{G}(\boldsymbol{T})\boldsymbol{\tau},} \tag{8.8}$$

with $\boldsymbol{G}$ the wrench basis of the manipulator, which depends on the pose $\boldsymbol{T}$ of the end-effector platform. (Hence, the forward position kinematics have to be solved before the forward force kinematics.)

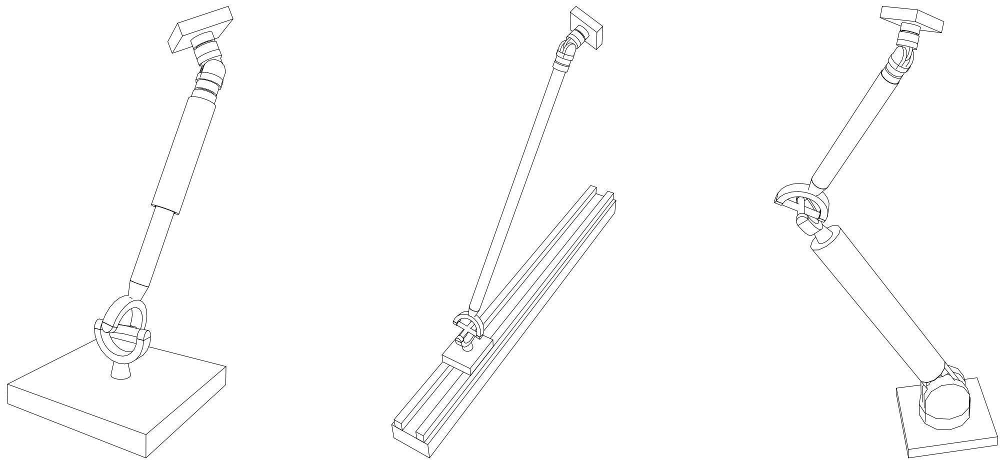


Figure 8.8: UPS, PUS and RUS "legs." "*R*" stands for *revolute* joint, "*P*" for *prismatic* joint, "*S*" for *spherical* joint and "*U*" for *universal* joint.

8.6.1 Partial wrenches for common "legs"

Figure 8.8 shows some examples of serial kinematic chains that are often used as "legs" in a parallel robot. The UPS is the "leg" structure for the Stewart-Gough platform, the PUS for the Hexaglide, [17], and the RUS for the HEXA design, [6, 37]. The partial wrench for all joints in these special chains can be found by inspection.

UPS chain The partial wrench for one of the three revolute joints in the spherical joint is the sum of:

- A moment orthogonal to the plane formed by the two other joints in the S joint. This moment doesn't have a component along the other revolute joints of the S joint. It will, however, cause components along the two revolute joints of the U joint.
- A force through the center of the spherical wrist, orthogonal to the axis of the P joint and the joint in the S joint for which the partial wrench is calculated. This force generates reaction moments in the U joint only. The magnitude of this force must be such that it compensates the influence of the moment mentioned above.

The partial wrench of each revolute joint in the U joint is a pure force determined geometrically by the following constraints:

- It intersects the center of the spherical joint.
- It is orthogonal to the P joint axis.
- It lies in the plane of this P joint axis and axis of the other revolute joint in the U joint.

Finally, the partial wrench for the P joint is a pure force along its axis. In most designs, only this P joint is actuated, such that finding the corresponding column in the wrench Jacobian $\boldsymbol{G}$ is very simple.

PUS chain The partial wrench for each revolute joint in the U and S joints is found exactly as in the case of a UPS structure. The partial wrench for the P joint is a pure force along the line through the centers of the U and S joints.

RUS chain The partial wrench for the R joint is a pure force along the line through the centers of the U and S joints. The partial wrench for a revolute joint in the S joint is the sum of:

- A moment orthogonal to the plane formed by the two other joints in the S joint. This moment doesn't have a component along the other revolute joints of the spherical wrist. It will, however, cause components along the two revolute a force through the center of the U joint
- A force through the center of the spherical wrist, orthogonal to the axis of the joint for which the partial wrench is calculated, and coplanar with the axis of the R joint. This force generates reaction moments in the U joint only. The magnitude of this force must be such that it compensates the influence of the moment mentioned above.

The partial wrench for the revolute joints in the U joint is a force along the line that intersects the center of the S joint, the axis of the R joint, and the axis of the other revolute joint in the U joint.

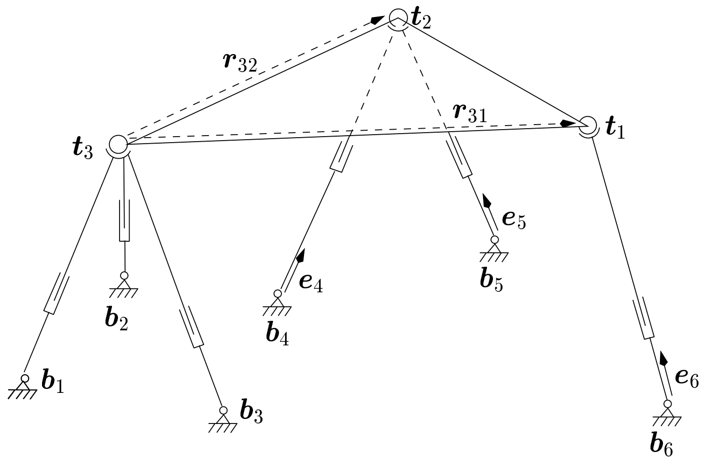


Figure 8.9: 321 structure, with notations. $\boldsymbol{t}_i$ is the point where i legs intersect. $\boldsymbol{e}_i$ is a unit vector along the ith leg.

8.6.2 Wrench basis for 321 structure

The Jacobian matrix $\boldsymbol{J}$ for the serial 321 structure was derived by inspection (Sect. 7.8.3), when using a reference frame with origin in the wrist centre point, i.e., the point where three of the joint axes intersect. Dually, the wrench basis $\boldsymbol{G}$ of the parallel 321 structure can be found by inspection, when using a reference frame with origin in the intersection point of the three prismatic legs, i.e., point $\boldsymbol{t}_3$ in Fig. 8.9. Because the partial wrench for a UPS leg is a pure force along its axis, $\boldsymbol{G}$ has the following form:

$$ {}_3\boldsymbol{G} = \begin{pmatrix} \boldsymbol{e}_1 & \boldsymbol{e}_2 & \boldsymbol{e}_3 & \boldsymbol{e}_4 & \boldsymbol{e}_5 & \boldsymbol{e}_6 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{r}_{32} \times \boldsymbol{e}_4 & \boldsymbol{r}_{32} \times \boldsymbol{e}_5 & \boldsymbol{r}_{31} \times \boldsymbol{e}_6 \end{pmatrix}. \tag{8.9} $$

Again, this matrix has a vanishing off-diagonal 3×3 submatrix.

8.7 Inverse velocity kinematics

Dual reasoning to the case of the inverse *force* kinematics for serial manipulators yields the inverse *velocity* kinematics ("IVK") of any parallel manipulator (with at least six six-jointed legs), i.e., the equality of the instantaneous power generated in joint space on the one hand, and in Cartesian space on the other hand, leads straightforwardly to the following "Jacobian transpose" equation:

$$ \boxed{\dot{\boldsymbol{q}} = (\widetilde{\boldsymbol{\Delta}\boldsymbol{G}})^T \mathbf{t}^{ee}.} \tag{8.10} $$

Again, a direct derivation of the same result exists:

Inverse velocity kinematics

Step 1 Let $\dot{\boldsymbol{l}}_i$ be the translational velocity of the end point of the ith leg, i.e., of the point connected to the end-effector platform. Place a reference frame $\{i\}$ with origin at the end point of leg i, and with its Z axis along $\boldsymbol{l}_i$. The unit vector $\boldsymbol{e}_z^i = \boldsymbol{l}_i / l_i$ in this direction is known from the IPK.

Step 2 The twist ${}_{bs}\mathbf{t}^{ee}$ of the end-effector is given with respect to the base frame $\{bs\}$, and with components expressed in this base frame. The second three-vector in this twist represents the translational velocity of the virtual point on the end-effector that instantaneously coincides with the origin of the base frame. What we need is the instantaneous translational velocity of the point that coincides with the origin of $\{i\}$. The transformation formula (6.25) in Sect. 6.6 yields:

$$ {}_i\mathbf{t}^{ee} = {}^{bs}_{i}\boldsymbol{S}\ {}_{bs}\mathbf{t}^{ee}, \quad \text{with} \quad {}^{bs}_{i}\boldsymbol{S} = \begin{pmatrix} {}^{bs}_{i}\boldsymbol{R} & \boldsymbol{0} \\ \left[{}_i\boldsymbol{p}^{i,bs}\right] {}^{bs}_{i}\boldsymbol{R} & {}^{bs}_{i}\boldsymbol{R} \end{pmatrix}. \tag{8.11} $$

The linear velocity component in the screw twist ${}_i\mathbf{t}^{ee}$ is the requested velocity, expressed in reference frame $\{i\}$.

Step 3 We know the third *column* of ${}^{i}_{bs}\boldsymbol{R}$, i.e., ${}_{bs}\boldsymbol{e}_z^i$. Hence, we know the third *row* of ${}^{bs}_{i}\boldsymbol{R} = {}^{i}_{bs}\boldsymbol{R}^T$. The vector $\boldsymbol{p}^{i_{ee},bs}$ is also known (Eq. (8.1) and Fig. 8.4):

$$ \boldsymbol{p}^{i_{ee},bs} = -(\boldsymbol{p}^{bs,ee} + \boldsymbol{p}^{ee,i_{ee}}) = -\boldsymbol{l}_i - \boldsymbol{p}^{bs,i_{bs}}. \tag{8.12} $$

Step 4 The velocity $\dot{\boldsymbol{l}}_i$ corresponds to the sixth component of ${}_i\mathbf{t}^{ee}$, i.e., the Z component of the translational velocity of the origin of $\{i\}$. To calculate this component, we need the sixth row of

${}^{bs}_{i}\boldsymbol{S}$. By definition, this sixth row equals the sixth column of ${}^{bs}_{i}\boldsymbol{S}^T$. The first three rows of this column are calculated from Eq. (8.11):

$$\begin{aligned}\left(\left[{}_{i}\boldsymbol{p}^{i_{ee},bs}\right]{}^{bs}_{i}\boldsymbol{R}\right)^T &= {}^{bs}_{i}\boldsymbol{R}^T\left[-{}_{i}\boldsymbol{p}^{i_{ee},bs}\right]\\ &= {}^{i}_{bs}\boldsymbol{R}\left[{}_{i}\boldsymbol{p}^{bs,i_{ee}}\right]\\ &= \left[{}_{bs}\boldsymbol{p}^{bs,i_{ee}}\right]{}^{i}_{bs}\boldsymbol{R}.\end{aligned}$$

So, the last column of this expression is

$$\left[{}_{bs}\boldsymbol{p}^{bs,i_{ee}}\right]{}_{bs}\boldsymbol{e}^{i}_{z} = {}_{bs}\boldsymbol{p}^{bs,i_{ee}} \times {}_{bs}\boldsymbol{e}^{i}_{z}. \tag{8.13}$$

The last three rows are the last column of ${}^{bs}_{i}\boldsymbol{R}^T$, which is the unit vector along the Z axis of frame $\{i\}$, expressed in frame $\{bs\}$: ${}_{bs}\boldsymbol{e}^{i}_{z}$.

The velocity $\dot{\boldsymbol{l}}_i$ (expressed in $\{i\}$) is then

$$\dot{\boldsymbol{l}}_i = \left({}_{i}\mathbf{t}^{ee}\right)_6 = \begin{pmatrix}{}_{bs}\boldsymbol{p}^{bs,i_{ee}} \times {}_{bs}\boldsymbol{e}^{i}_{z}\\ {}_{bs}\boldsymbol{e}^{i}_{z}\end{pmatrix}^T {}_{bs}\mathbf{t}^{ee} = \left(\widetilde{\boldsymbol{\Delta}}\begin{pmatrix}{}_{bs}\boldsymbol{e}^{i}_{z}\\ {}_{bs}\boldsymbol{p}^{bs,i_{ee}} \times {}_{bs}\boldsymbol{e}^{i}_{z}\end{pmatrix}\right)^T {}_{bs}\mathbf{t}^{ee}. \tag{8.14}$$

Step 5 The six-vector in Eq. (8.14) between the $\widetilde{\boldsymbol{\Delta}}$ and the end-effector twist ${}_{bs}\mathbf{t}^{ee}$ is the screw that represents a pure force along $\boldsymbol{l}_i$. For a Stewart-Gough leg along this direction $\boldsymbol{l}_i$, this corresponds to the definition of the actuated joint's partial wrench. Hence, the ith column of the "Jacobian transpose" equation (8.10) is found.

For a HEXA leg, a similar reasoning can be followed, but now the Z^i-axis of the frame $\{i\}$ is to be placed along the direction of the top link of the leg (l^t_i in Fig. 8.7). A force along this direction is also the partial wrench for the actuated revolute joint in the HEXA leg, since it generates no torques in any of the other joints of the leg.

In the equation above, we omitted the trailing "bs" subscript for the twist $\mathbf{t}^{ee}$, since this equation is valid for all reference frames (if, of course, the Jacobian matrix is expressed with respect to this same reference frame!).

8.8 Forward position kinematics

The forward position kinematics ("FPK") solves the following problem: *Given the vector of leg positions* $\boldsymbol{q} = (q_1 \ \dots \ q_n)^T$*, what is the corresponding end-effector pose?* This problem is in general highly nonlinear, and is dual to the IPK of serial manipulators. FPK algorithms have to solve a 40th degree polynomial for a general parallel structure, [32]; this reduces to a 16th degree polynomial for the special designs of the Stewart-Gough platform, [14, 29]. The following subsections discuss how to solve the IPK problem *numerically*, but also which designs allow *closed-form* solutions.

Fact-to-Remember 59 (Forward kinematics)
No closed-form solution exists for the forward position kinematics of a general parallel structure, and to one set of joint positions correspond many end-effector poses ("configurations," or "assembly modes").

8.8.1 General parallel structure

The numerical procedure, [10, 36], runs along the same lines as the IPK for a serial robot (Sect. 7.9):

Numerical FPK

Step 0 Start with an *estimate* $\widehat{\boldsymbol{T}}_0$ of the end-effector pose that corresponds to the vector $\boldsymbol{q}$ of six leg lengths. $\widehat{\boldsymbol{T}}_0$ is the first in a series of iterations, so initialise this iteration as

$$i = 0, \quad \widehat{\boldsymbol{T}}_i = \widehat{\boldsymbol{T}}_0. \tag{8.15}$$

Step 1 Use the inverse position kinematics to calculate (i) the joint positions $\hat{\boldsymbol{q}}_i$ that correspond to the estimate $\widehat{\boldsymbol{T}}_i$, and (ii) the error leg length vector $\Delta\boldsymbol{q}_i$:

$$\Delta\boldsymbol{q}_i = \boldsymbol{q} - \hat{\boldsymbol{q}}_i. \tag{8.16}$$

Step 2 Calculate the wrench basis $\boldsymbol{G}_i$ that corresponds to the latest estimate $\widehat{\boldsymbol{T}}_i$, by means of the IPK routine of Sect. 8.5 and the partial wrench of the actuated joints in each leg.

Step 4 Use the inverse of the "Jacobian transpose" equation (8.10) to calculate a new infinitesimal update $\mathbf{t}_{\Delta,i+1}$ for the current estimate $\widehat{\boldsymbol{T}}_i$ of the end-effector pose:

$$\mathbf{t}_{\Delta,i+1} = (\widetilde{\boldsymbol{\Delta}}\boldsymbol{G}_i)^{-T}\Delta\boldsymbol{q}_i. \tag{8.17}$$

Step 5 Update $\widehat{\boldsymbol{T}}_i$:

$$\widehat{\boldsymbol{T}}_{i+1} = \widehat{\boldsymbol{T}}_i\boldsymbol{T}(\mathbf{t}_{\Delta,i+1}). \tag{8.18}$$

$\boldsymbol{T}(\mathbf{t}_{\Delta,i+1})$ is the homogeneous transformation matrix that corresponds to the infinitesimal displacement twist $\mathbf{t}_{\Delta,i+1}$, Eq. (6.18).

Step 6 The iteration stops if $\Delta\boldsymbol{q}_i$ is "small enough."

Step 4 of this algorithm requires the inverse of the wrench basis $\boldsymbol{G}$. Hence, this appoach can only be applied unambiguously to manipulators with *six* actuated joints. As in all numerical algorithms, a good start configuration is required, such that the Newton-Raphson type of iteration used in the numerical procedure can converge to the desired solution. Solving the IPK of a serial robot (Sect. 7.9.1) requires a *joint space* start configuration; the FPK for parallel robots, however, requires a *Cartesian space* initial estimate [24], i.e., an estimate $\widehat{\boldsymbol{T}}_0$ of the end-effector pose. Due to the limited workspace of parallel manipulators, the "zero configuration" end-effector pose of the manipulator could serve as an appropriate initial estimate. Moreover, one often is only interested in the solution with the same configuration as this zero configuration. Note however that no strict definition exists for the zero configuration.

8.8.2 Closed-form FPK for 321 structure

Only a very limited number of fully parallel kinematic designs with a closed-form FPK solution is known. The 321 structure is one example; the other examples are:

1. *Linearly dependent base and end-effector platforms.* A closed-form solution to the FPK exists if the coordinates of the leg pivot points on the end-effector are particular linear combinations of the coordinates of the pivot points on the base, [7, 39]. A special case occurs when both platforms are *similar hexagons* [21, 34, 38], i.e., the pivot points on both platforms lie on polygons that are equal up to a scaling factor, Fig. 8.10. (This design seems attractive at first sight, but it turns out to have a very bad singularity manifold, [8].)

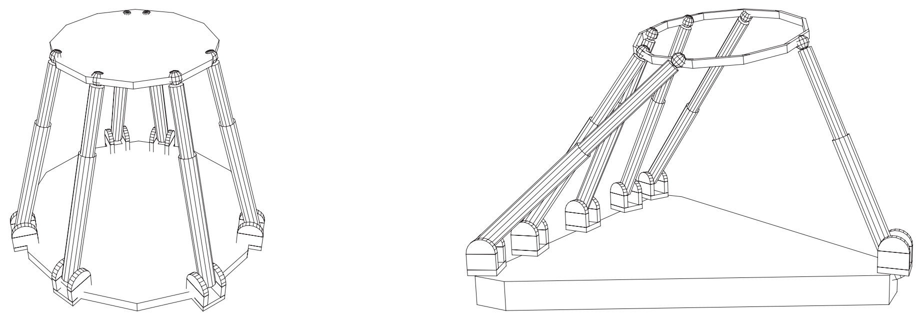


Figure 8.10: Two parallel manipulator designs that allow a closed-form forward kinematics solution: base and end-effector platforms are *similar hexagons* (left), or five leg pivot points are *collinear* (right).

2. *Five collinear pivot points.* A closed-form solution to the FPK exists if the base and/or end-effector contains five *collinear* pivot points, [40] (Fig. 8.10).

The following paragraphs give the full mathematical treatment for the the 321 design only. As mentioned before, this 321 design has a tetrahedral substructure, formed by three legs (Fig. 8.5). The following sides of this tetrahedron are known (Fig. 8.11): the lengths between the top T on the end-effector and the points P, Q, and R on the base (since these are *measured*), and the lengths of the sides PQ, QR and RP (since these are design *constants*). The following paragraphs show how to find the *position* coordinates of the top point T.

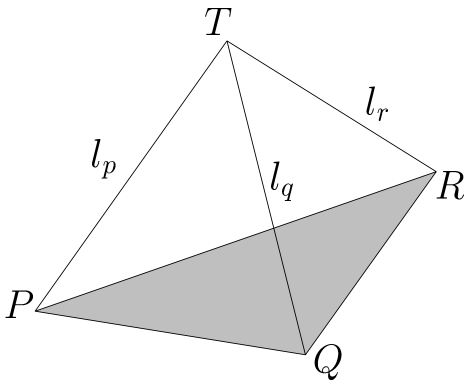


Figure 8.11: Tetrahedral substructure of the 321 structure.

Let $\boldsymbol{p} = (p_x\ p_y\ p_z)^T, \boldsymbol{q} = (q_x\ q_y\ q_z)^T$ and $\boldsymbol{r} = (r_x\ r_y\ r_z)^T$ be the coordinate three-vectors (with respect to an arbitrary world reference frame) of the base points P, Q and R, respectively. Let $\boldsymbol{t} = (t_x\ t_y\ t_z)^T$ be the (unknown) coordinate three-vector of the top T; and let $\boldsymbol{s} = (s_x\ s_y\ s_z)^T$ be the vector $\boldsymbol{t} - \boldsymbol{p}$ from the base point P to the top T. If l_p is the length of this side PT, then

$$s_x^2 + s_y^2 + s_z^2 = l_p^2. \tag{8.19}$$

Let $\boldsymbol{a} = (a_x\ a_y\ a_z)^T$ be the (known) vector from P to Q (i.e., $\boldsymbol{a} = \boldsymbol{q} - \boldsymbol{p}$), with length a, and $\boldsymbol{b} = (b_x\ b_y\ b_z)^T$ the (known) vector from P to R (i.e., $\boldsymbol{b} = \boldsymbol{r} - \boldsymbol{p}$), with length b. Let $\boldsymbol{c} = (c_x\ c_y\ c_z)^T$ be the (unknown) vector from Q to T; its length l_q is known. Let $\boldsymbol{d} = (d_x\ d_y\ d_z)^T$ be the (unknown) vector from R to T; its length l_r is known.

Hence

$$\boldsymbol{a} - \boldsymbol{s} = -\boldsymbol{c},$$
$$\boldsymbol{b} - \boldsymbol{s} = -\boldsymbol{d}.$$

Taking the dot products of these identities with themselves gives

$$a_x s_x + a_y s_y + a_z s_z = (l_p^2 + a^2 - l_q^2)/2, \tag{8.20}$$
$$b_x s_x + b_y s_y + b_z s_z = (l_p^2 + b^2 - l_r^2)/2. \tag{8.21}$$

The right-hand sides of these equations are completely known, as well as the scalars a_x, a_y, a_z, b_x, b_y and b_z. So, in general, one can express s_x and s_y in terms of s_z, by elimination from Eqs (8.20) and (8.21). Equation (8.19) then yields a quadratic equation in s_z, with two solutions. Backsubstitution of the result in (8.20) and (8.21) yields s_x and s_y. The coordinate three-vector $\boldsymbol{t}$ of the top T is then simply $\boldsymbol{t} = \boldsymbol{p} + \boldsymbol{s}$. If the quadratic equation in s_z has only complex roots, the lengths of the sides of the tetrahedron are not compatible with the dimensions of the base, i.e., the tetrahedron cannot be "closed" with the given lengths. Note that (i) the coordinates of the top of a tetrahedron are found from *linear* and *quadratic* equations only, and (ii) the two solutions correspond to reflections of the top point T about the plane PQR.

Until now, the position of only one point of the top platform is known. However, finding the positions of the other points is done by repeating the tetrahedron algorithm above: the point on the top platform that is connected to two legs also forms a tetrahedron with known lengths of these two legs as well as of the length of the connection to T. Hence, the position of this point can be found. A similar reasoning holds for the third point. Since we know the *position* coordinates of three points on the end-effector platform, we can derive its *orientation*, [1]. Conceptually the simplest way to do this is to apply the tetrahedron algorithm four more times: the three leg pivot points on the top platform are the tetrahedron base points, and the four vertices of the end-effector frame (i.e., its origin and the end-points of the unit vectors along X, Y and Z) are the tetrahedron top points. The rotation matrix of this end-effector frame is then straightforwardly found by subtracting the coordinates of the frame origin from the coordinates of the end-points of the frame unit vectors.

The tetrahedron procedure has been applied to three tetrahedrons in the 321 structure. Each application gives two different configurations. So, the 321 parallel manipulator has *eight* forward position kinematics solutions.

8.8.3 Closed-form FPK: sensing redundancy

Another approach to construct closed-form FPK solutions exists, and it works with all possible kinematic designs: add extra sensors. Two complementary approaches are known, [4, 16, 20, 21, 25, 26, 30]:

1. Add extra non-actuated legs whose lengths can be measured. In this way, it is, for example, possible to (i) construct a six-legs substructure that has one of the above-mentioned closed-form architectures, or (ii) to build tetrahedral substructures as in Figure 8.11. Each tetrahedron unambiguously determines the *position* of one point of the moving platform.

2. Add position sensors to one or more of the passive joints in the existing legs. In this way, it is for example possible to measure the full position vector $\boldsymbol{l}_i$ of the ith leg, instead of only its length q_i.

The drawbacks of adding more sensors to the manipulator are: (i) the system becomes more expensive, (ii) the extra sensors take up space which decreases the already limited free workspace of the manipulator even further, and (iii) due to measurement noise, manufacturing tolerances, etc., the FPK results can become inconsistent (since one gets more than six measurements for only six independent variables). On the other hand, extra sensors facilitate the *calibration* of the robot.

8.9 Forward velocity kinematics

The forward velocity kinematics ("FVK") of a parallel robot are dual to the inverse velocity kinematics of a serial robot, Sect. 7.10.

8.9.1 General parallel robots

The obvious numerical technique to solve the FVK inverts the "Jacobian transpose" equation (8.10):

$$\boxed{\mathbf{t}^{ee} = (\widetilde{\boldsymbol{\Delta}}\boldsymbol{G})^{-T}\dot{\boldsymbol{q}}.} \tag{8.22}$$

This approach requires the FPK solution, in order to construct the wrench basis $\boldsymbol{G}$.

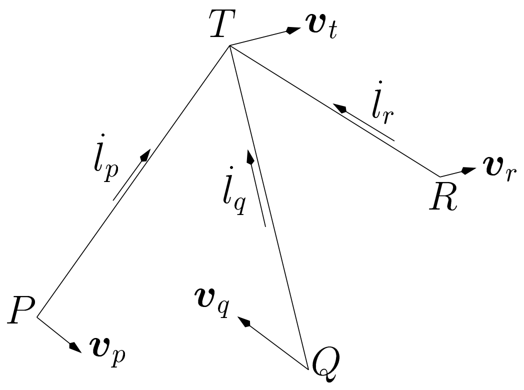


Figure 8.12: Velocities in a tetrahedron of the 321 structure.

8.9.2 Closed-form FVK for 321 structure

The wrench basis $\boldsymbol{G}$ of Eq. (8.9) has a 3×3 submatrix of zeros, and hence it is invertible symbolically as:

$$\boldsymbol{G} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{C} \end{pmatrix} \Rightarrow \boldsymbol{G}^{-1} = \begin{pmatrix} \boldsymbol{A}^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{C}^{-1} \\ \boldsymbol{0} & \boldsymbol{C}^{-1} \end{pmatrix}. \tag{8.23}$$

And the matrices $\boldsymbol{A}^{-1}$ and $\boldsymbol{C}^{-1}$ are also easy to find by inspection. For example, the first *row* of $\boldsymbol{A}^{-1}$ consists of the vector orthogonal to the second and third columns of $\boldsymbol{A}$, and having a unit scalar product with the first column of $\boldsymbol{A}$. Similarly for the other rows, and also for the matrix $\boldsymbol{C}^{-1}$.

FVK with tetrahedron algorithm. As for the forward position kinematics, the forward velocity kinematics of the 321 design can also be found directly, as an iterative solution of a tetrahedral substructure. Indeed, assume now that also the *velocity three-vectors* $\boldsymbol{v}_p, \boldsymbol{v}_q$ and $\boldsymbol{v}_r$ of the base points are given, Fig. 8.12. The instantaneous velocity $\boldsymbol{v}_t$ of the top is the sum of the top point velocities generated by each of the base point velocities individually, keeping the two other base points motionless. Hence, assume $\boldsymbol{v}_q = \boldsymbol{v}_r = 0$ and $\boldsymbol{v}_p \neq 0$. Since Q and R do not move and the lengths l_q and l_r do not change, the top point can only move along a line that is perpendicular to the direction QT as well as to the direction RT. Hence, the unit direction vector $\boldsymbol{e}$ of the top's instantaneous velocity is found from:

$$0 = \boldsymbol{e} \cdot (\boldsymbol{t} - \boldsymbol{q}), \tag{8.24}$$

$$0 = \boldsymbol{e} \cdot (\boldsymbol{t} - \boldsymbol{r}), \tag{8.25}$$

$$1 = \boldsymbol{e} \cdot \boldsymbol{e}, \tag{8.26}$$

with $\boldsymbol{q}, \boldsymbol{r}$ and $\boldsymbol{t}$ the position three-vectors of the points Q, R and T. This set of equations is similar to Eqs (8.19)–(8.21), and hence again requires only linear and quadratic equations. At this point, we have a known velocity $\boldsymbol{v}_p$ of one end of the tetrahedron side PT, and an unknown velocity $\boldsymbol{v}_t$ (i.e., unknown magnitude v_t but known direction $\boldsymbol{e}$) at the other end of the rigid link PT. Hence, they must be such that the length l^t of the leg does not change. This means that both velocities have the same projection along the direction of the leg. This direction is instantaneously given by the vector $\boldsymbol{t} - \boldsymbol{p}$. Hence

$$0 = \frac{\mathrm{d}l^t}{\mathrm{d}t} = \boldsymbol{v}_p \cdot (\boldsymbol{t} - \boldsymbol{p}) - v_t \boldsymbol{e} \cdot (\boldsymbol{t} - \boldsymbol{p}). \tag{8.27}$$

This equation gives v_t since $\boldsymbol{p}, \boldsymbol{v}_p, \boldsymbol{t}$, and the direction $\boldsymbol{e}$ of $\boldsymbol{v}_t$ are known. Repeating the same reasoning for the similar cases $\{\boldsymbol{v}_r = \boldsymbol{v}_p = 0, \boldsymbol{v}_q \neq 0\}$ and $\{\boldsymbol{v}_p = \boldsymbol{v}_q = 0, \boldsymbol{v}_r \neq 0\}$, and summing the result, yields the total instantaneous velocity of the top. And as before, this tetrahedron algorithm can be applied iteratively to all three points on the end-effector platform. This yields the velocity of these three points. From the velocity of three points, one can find the angular velocity of the platform; see [2, 11, 33].

8.10 Singularities

Equation (8.22) shows that the forward velocity kinematics of a fully parallel manipulator becomes singular if the wrench basis matrix $\boldsymbol{G}$ becomes singular. This means that at most only five of its six screws are independent, and hence one or more force resistance degrees of freedom are lost. In other words, the manipulator has *gained* one or more passive instantaneous motion degrees of freedom. Note again that a singularity occurs in general not just in one single position of the robot, but in a continuously connected *manifold.*

8.10.1 Singularities for the 321 structure

The singularities of the 321 structure are easily found from Eq. (8.9): $\det(\boldsymbol{G}) = 0 \Leftrightarrow \det(\boldsymbol{e}_1\ \boldsymbol{e}_2\ \boldsymbol{e}_3) = 0$ and $\det(\boldsymbol{r}_{32} \times \boldsymbol{e}_4\ \boldsymbol{r}_{32} \times \boldsymbol{e}_5\ \boldsymbol{r}_{31} \times \boldsymbol{e}_6) = 0$. Hence, the "321" structure has *three singularities* (or rather, singularity manifolds):

"3"-singularity : the "3"-point $\boldsymbol{t}_3$ lies in the plane of the "base" formed by $\boldsymbol{b}_1, \boldsymbol{b}_2$ and $\boldsymbol{b}_3$. This means that the three unit vectors $\boldsymbol{e}_1, \boldsymbol{e}_2$ and $\boldsymbol{e}_3$ along the first three legs have become linearly dependent: the *forces* generated by the first *three* legs form only a *two*-dimensional space. In practice, the platform will jump unpredictably to one of two sub-configurations, "3-up" or "3-down."

"2"-singularity : the "2"-point $\boldsymbol{t}_2$ is coplanar with points $\boldsymbol{t}_3$, $\boldsymbol{b}_4$ and $\boldsymbol{b}_5$, hence $\det(\boldsymbol{r}_{32} \times \boldsymbol{e}_4\ \boldsymbol{r}_{32} \times \boldsymbol{e}_5\ \boldsymbol{r}_{31} \times \boldsymbol{e}_6) = 0$, because the first two columns are dependent. This singularity separates two sub-configurations, "2-up" and "2-down." Its physical interpretation is the same as in the "3"-singularity.

"1"-singularity : the "1"-point $\boldsymbol{t}_1$ is coplanar with points $\boldsymbol{t}_3$, $\boldsymbol{t}_2$ and $\boldsymbol{b}_6$. Hence, $\det(\boldsymbol{r}_{32} \times \boldsymbol{e}_4\ \boldsymbol{r}_{32} \times \boldsymbol{e}_5\ \boldsymbol{r}_{31} \times \boldsymbol{e}_6) = 0$, because all three vectors in this determinant are orthogonal to the same vector $\boldsymbol{r}_{32}$ and so must be dependent. Any force generated in this plane by leg 6 is a linear combination of the forces in this plane generated by the other legs. This singularity again separates two sub-configurations, "1-up" and "1-down." Its physical interpretation is again the same as in the "3" and "2"-singularities.

Hence, the "321" structure has *three singularities* and *eight configurations*, each of these eight labelled by a binary choice of sub-configuration (e.g., "3-up, 2-down, 1-up"). (The names of the singularities and configurations are not standardized!)

8.11 Redundancy

A parallel manipulator is called redundant if it has more than six actuated joints. Such a design could be advantageous for several reasons, such as:

1. The manipulator keeps full actuation capability around singularities.
2. More legs make the load to be moved by every leg smaller, hence heavier loads can be carried, and with higher speeds.
3. The robot can move between different assembly modes, which increases its workspace.

On the other hand, extra legs increase the collision danger and the cost. Dually to a redundant serial manipulator, a redundant parallel manipulator has no choice in optimizing *leg motion* (the motion of all legs are still coupled by the closure equations (8.1) and their time derivatives!) but it can optimise the *force distribution* in its legs. The reasoning for redundant serial manipulators (Sect. 7.14) can be repeated here, with $\dot{\boldsymbol{q}}$ replaced by $\boldsymbol{\tau}$, and $\mathbf{t}^{ee}$ by $\mathbf{w}^{ee}$:

1. A null space of leg forces exists, i.e., there is an infinite set of leg forces that cause no end-effector wrench, but cause *internal forces* in the platforms:

$$\boxed{\text{Null}\,(\boldsymbol{G}) = \left\{\boldsymbol{\tau}^N \mid \boldsymbol{G}\,\boldsymbol{\tau}^N = \mathbf{0}\right\}.} \tag{8.28}$$

   This null space depends on the current platform pose.

2. If a wrench $\mathbf{w}^{ee}$ acts on the end-effector, it can be statically resisted by a set of leg forces that *minimises* the static *deformation* of the legs, and hence maximises the position accuracy. The optimization criterion is as follows:

$$\begin{cases} \min\limits_{\boldsymbol{\tau}} \quad P = \dfrac{1}{2}\boldsymbol{\tau}^T \boldsymbol{K}^{-1}\boldsymbol{\tau}, \\ \text{such that} \quad \mathbf{w}^{ee} = \boldsymbol{G}\,\boldsymbol{\tau}. \end{cases} \tag{8.29}$$

   The positive scalar P is the potential energy stored in the manipulator, and generated by deforming the compliance k_i^{-1} of each leg by the force τ_i in the leg; the matrix $\boldsymbol{K}$ is the *joint space stiffness matrix*, i.e., the diagonal matrix diag$(k_1, \ldots, k_n)$, if there are n legs in the manipulator. The same reasoning as for a serial redundant manipulator applies, with $\dot{\boldsymbol{q}}$ replaced by $\boldsymbol{\tau}$, $\boldsymbol{J}$ by $\boldsymbol{G}$, $\boldsymbol{M}$ by $\boldsymbol{K}^{-1}$, and $\mathbf{t}^{ee}$ by $\mathbf{w}^{ee}$. Hence, the optimal solution is given by the dual of Eq. (7.65)

$$\boldsymbol{\tau} = \boldsymbol{K}\boldsymbol{G}^T \left(\boldsymbol{G}\boldsymbol{K}\boldsymbol{G}^T\right)^{-1} \mathbf{w}^{ee} \tag{8.30}$$

$$\triangleq \boldsymbol{G}_K^{\dagger}\mathbf{w}^{ee}. \tag{8.31}$$

   $\boldsymbol{G}_K^{\dagger}$ is the weighted pseudo-inverse, with stiffness matrix $\boldsymbol{K}$ acting as weighting matrix on the space of joint forces.

8.11.1 Summary of dualities

	SERIAL	PARALLEL
FPK	easy, unique	difficult multiple solutions
IPK	difficult multiple solutions	easy, unique
FVK	always defined column of $\boldsymbol{J}$ = twist of joint axis $\mathbf{t}^{ee} = \boldsymbol{J}\dot{\boldsymbol{q}}$	redundancy, singularities $\mathbf{t}^{ee} = (\widetilde{\boldsymbol{\Delta}}\boldsymbol{G})^{-T}\dot{\boldsymbol{q}}$
FFK	redundancy, singularities $\mathbf{w}^{ee} = (\widetilde{\boldsymbol{\Delta}}\boldsymbol{J})^{-T}\boldsymbol{\tau}$	always defined column of $\boldsymbol{G}$ is partial wrench of joint $\mathbf{w}^{ee} = \boldsymbol{G}\boldsymbol{\tau}$
IVK	redundancy, singularities $\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}\mathbf{t}^{ee}$	always defined $\dot{\boldsymbol{q}} = (\widetilde{\boldsymbol{\Delta}}\boldsymbol{G})^{T}\mathbf{t}^{ee}$
IFK	always defined $\boldsymbol{\tau} = (\widetilde{\boldsymbol{\Delta}}\boldsymbol{J})^{T}\mathbf{w}^{ee}$	redundancy, singularities $\boldsymbol{\tau} = \boldsymbol{G}^{-1}\mathbf{w}^{ee}$
Singularities	rank($\boldsymbol{J}$) drops loose active motion degree of freedom gain passive force degree of freedom	rank($\boldsymbol{G}$) drops loose active force degree of freedom gain passive motion degree of freedom
Redundancy	$\boldsymbol{J}$ has null space: motion distribution $\dot{\boldsymbol{q}} = \boldsymbol{J}^{\dagger}_{M^{-1}}\mathbf{t}^{ee}$	$\boldsymbol{G}$ has null space: force distribution $\boldsymbol{\tau} = \boldsymbol{G}^{\dagger}_{K}\mathbf{w}^{ee}$
Closed-form	321: intersecting revolute joints	321: intersecting prismatic joints

References for this Chapter

[1] J. Angeles. Automatic computation of the screw parameters of rigid-body motions. Part I: Finitely-separated positions. *Trans. ASME J. Dyn. Systems Meas. Control*, 108:32–38, 1986.

[2] J. Angeles. Automatic computation of the screw parameters of rigid-body motions. Part II: Infinitesimally-separated positions. *Trans. ASME J. Dyn. Systems Meas. Control*, 108:39–43, 1986.

[3] J. Angeles. *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms.* Mechanical Engineering Series. Springer-Verlag, 1997.

[4] L. Baron and J. Angeles. The decoupling of the direct kinematics of parallel manipulators using redundant sensors. In *IEEE Int. Conf. Robotics and Automation*, pages 974–979, San Diego, CA, 1994.

[5] D. Bernier, J.-M. Castelain, and X. Li. A new parallel structure with six degrees of freedom. In *Ninth World Congress on the Theory of Machines and Mechanisms*, pages 8–12, Milano, Italy, 1995.

[6] H. Bruyninckx. The 321-HEXA: A fully-parallel manipulator with closed-form position and velocity kinematics. In *IEEE Int. Conf. Robotics and Automation*, pages 2657–2662, Albuquerque, NM, 1997.

[7] H. Bruyninckx and J. De Schutter. A class of fully parallel manipulators with closed-form forward position kinematics. In J. Lenarčič and V. Parenti-Castelli, editors, *Recent Advances in Robot Kinematics*, pages 411–419, Portorož-Bernardin, Slovenia, 1996.

[8] H. Bruyninckx and J. De Schutter. Comments on "Closed Form Forward Kinematics Solution to a Class of Hexapod Robots". *IEEE Trans. Rob. Automation*, 15, 1999. Submitted.

[9] R. Clavel. Delta, a fast robot with parallel geometry. In *Int. Symp. Industrial Robots*, pages 91–100, Lausanne, Switzerland, 1988.

[10] J. E. Dieudonne, R. V. Parrish, and R. E. Bardusch. An actuator extension transformation for a motion simulator and an inverse transformation applying Newton-Raphson's method. Technical Report NASA TN D-7067, NASA Langley Research Center, Hampton, VA, 1972.

[11] R. G. Fenton and R. A. Willgoss. Comparison of methods for determining screw parameters of infinitesimal rigid body motion from position and velocity data. *Trans. ASME J. Dyn. Systems Meas. Control*, 112:711–716, 1990.

[12] C. M. Gosselin and M. Gagné. A closed-form solution for the direct kinematics of a special class of spherical three-degree-of-freedom parallel manipulators. In J.-P. Merlet and B. Ravani, editors, *Computational Kinematics '95*, pages 231–240, Dordrecht, 1995. Kluwer Academic Publishers.

[13] V. E. Gough and S. G. Whitehall. Universal tyre test machine. In *Proc. 9th Int. Tech. Congress FISITA*, pages 117–137, 1962.

[14] M. Griffis and J. Duffy. A forward displacement analysis of a class of Stewart platforms. *J. Robotic Systems*, 6(6):703–720, 1989.

[15] G. J. Hamlin. *Tetrobot: A Modular System for Reconfigurable Parallel Robots.* PhD thesis, Rensselaer Polytechnic Institute, Troy, NY, 1996.

[16] K. Han, W. Chun, and Y. Youm. New resolution scheme of the forward kinematics of parallel manipulators using extra sensors. *Trans. ASME J. Mech. Design*, 118:214–219, 1996.

[17] M. Honegger, A. Codourey, and E. Burdet. Adaptive control of the Hexaglide, a 6 dof parallel manipulator. In *IEEE Int. Conf. Robotics and Automation*, pages 543–548, Albuquerque, NM, 1997.

[18] K. H. Hunt and P. R. McAree. The octahedral manipulator: Geometry and mobility. *Int. J. Robotics Research*, 17(8):868–885, 1998.

[19] M. Husain and K. J. Waldron. Direct position kinematics of the 3-1-1-1 Stewart platforms. *Trans. ASME J. Mech. Design*, 116:1102–1107, 1994.

[20] H. Inoue, Y. Tsusaka, and T. Fukuizumi. Parallel manipulator. In *Third International Symposium on Robotics Research*, pages 321–327, Gouvieux, France, 1985.

[21] H.-Y. Lee and B. Roth. A closed-form solution of the forward displacement analysis of a class of in-parallel mechanisms. In *IEEE Int. Conf. Robotics and Automation*, pages 720–724, Atlanta, GA, 1993.

[22] P. R. McAree and R. W. Daniel. A fast, robust solution to the Stewart platform forward kinematics. *J. Robotic Systems*, 13(7):407–427, 1996.

[23] H. McCallion and P. D. Truong. The analysis of a six-degree-of-freedom work station for mechanised assembly. In *Proc. 5th World Congress on Theory of Machines and Mechanisms*, pages 611–616, Montréal, Canada, 1979.

[24] J.-P. Merlet. Parallel manipulators: Part I: Theory; design, kinematics, dynamics and control. Technical Report 646, INRIA, Sophia Antipolis, France, 1987.

[25] J.-P. Merlet. Closed-form resolution of the direct kinematics of parallel manipulators using extra sensor data. In *IEEE Int. Conf. Robotics and Automation*, pages 200–204, Atlanta, GA, 1993.

[26] R. Nair and J. H. Maddocks. On the forward kinematics of parallel manipulators. *Int. J. Robotics Research*, 13(2):171–188, 1994.

[27] P. Nanua. Direct kinematics of parallel mechanisms. Master's thesis, Ohio State University, Ohio, 1988.

[28] P. Nanua and K. J. Waldron. Direct kinematic solution of a special parallel robot structure. In A. Morecki, G. Bianchi, and K. Jaworek, editors, *Proc. 8th CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators*, pages 134–142, Warsaw, Poland, 1990.

[29] P. Nanua, K. J. Waldron, and V. Murthy. Direct kinematic solution of a Stewart platform. *IEEE Trans. Rob. Automation*, 6(4):438–444, 1990.

[30] V. Parenti-Castelli and R. Di Gregorio. Real-time forward kinematics of the general Gough-Stewart platform using two additional rotary sensors. In *Mechatronics '96*, volume 1, pages 113–118, Guimarães, Portugal, 1996.

[31] F. Pierrot, A. Fournier, and P. Dauchez. Towards a fully-parallel 6 dof robot for high-speed applications. In *IEEE Int. Conf. Robotics and Automation*, pages 1288–1293, Sacramento, CA, 1991.

[32] M. Raghavan. The Stewart platform of general geometry has 40 configurations. *Trans. ASME J. Mech. Design*, 115:277–282, 1993.

[33] H. J. Sommer, III. Determination of first and second order instant screw parameters from landmark trajectories. *Trans. ASME J. Mech. Design*, 114:274–282, 1992.

[34] S. V. Sreenivasan, K. J. Waldron, and P. Nanua. Closed-form direct displacement analysis of a 6-6 Stewart platform. *Mechanism and Machine Theory*, 29(6):855–864, 1994.

[35] D. Stewart. A platform with six degrees of freedom. *Proc. Instn Mech. Engrs*, 180-1(15):371–386, 1965.

[36] K. Sugimoto. Kinematic and dynamic analysis of parallel manipulators by means of motor algebra. *Trans. ASME J. Mech. Transm. Automation Design*, 109:3–7, 1987.

[37] M. Uchiyama, K. Iimura, F. Pierrot, P. Dauchez, K. Unno, and O. Toyama. A new design of a very fast 6-DOF parallel robot. In *Int. Symp. Industrial Robots*, pages 771–776, Barcelona, Spain, 1992.

[38] G. Wang. Forward displacement analysis of a class of the 6-6 Stewart platforms. In *Proc. 1992 ASME Design Technical Conferences–22nd Biennal Mechanisms Conference*, pages 113–117, Scottsdale, AZ, 1992.

[39] J. Yang and Z. J. Geng. Closed form forward kinematics solution to a class of Hexapod robots. *IEEE Trans. Rob. Automation*, 14(3):503–508, 1998.

[40] C.-D. Zhang and S.-M. Song. Forward kinematics of a class of parallel (Stewart) platforms with closed-form solutions. *J. Robotic Systems*, 9(1):93–112, 1992.

Chapter 9

Mobile robot kinematics

9.1 Introduction

This Chapter treats *mobile robots*, i.e., devices such as unicycles, bicycles, cars, and, especially, the mobile devices that have two independently driven wheels on one axle and one or more passive support wheels on a second axle (Fig. 9.1), [15, 18]. This text calls them "*differentially-driven*" or "*caster*" robots. A caster wheel (or "castor" wheel) is a wheel mounted in a swivel frame, and used for supporting furniture, trucks, portable machines, etc. The caster wheel is *not actuated*. This text only considers mobile robots that move over a *plane*. Hence, they have two translational and one rotational degree of freedom; the rotation axis is perpendicular to the translations. This *configuration* space is nothing else but SE(2), Sect. 3.7. The *joint space* for a car-like robot is one-dimensional (turning the steering wheel does not *move* the robot!), but it is two-dimensional for a differentially-driven robot.

Mobile robots, at first sight, are rather different in nature from the serial and parallel manipulators of the previous Chapters. However, this Chapter will highlight many similarities, such that no new concepts are needed for a comprehensive treatment of mobile robot kinematics.

Nonholonomic constraint. The common characteristic of mobile robots is that they cannot be given a velocity which is *transversal* to the axle of their wheels. A differentially-driven robot has one such constraint (the caster wheels are mounted on a swivel and hence give no constraint); bicycles and cars have two constraints: one on the front wheel axle and one on the rear wheel axle. These constraits are *nonholonomic* constraints on the *velocity* of the robots, [10, 13], Sect. 7.14, i.e., they cannot be integrated to give a constraint on the robots' Cartesian *pose*. This means that the vehicle cannot move transversally *instantaneously*, but it can reach any position and orientation by moving backward and forward while turning appropriately. Parking your car is a typical example of this maneuver phenomenon. The nonholonomic constraints reduce the mobile robot's instantaneous velocity degrees of freedom, and hence most robots have only *two* actuated joints:

1. The two *driven wheels* in the case of a differentially-driven robot.
2. The *driven* wheels (driven by only *one*motor!), and the *steering* wheel of a car-like mobile robot. Only the driving speed is an *instantaneous* (or "first order") degree of freedom; the speed is only of *second order*, since by itself it does not generate a motion of the mobile robot.

Differentially-driven robots can turn "on the spot," while car-like robots cannot. Note the following difference between mobile robots on the one hand, and serial and parallel robots on the other hand: the angles of the wheel joints don't tell you where the vehicle is in its configuration space, and vice versa. This means that the *position* kinematics of mobile robots are not uniquely defined.

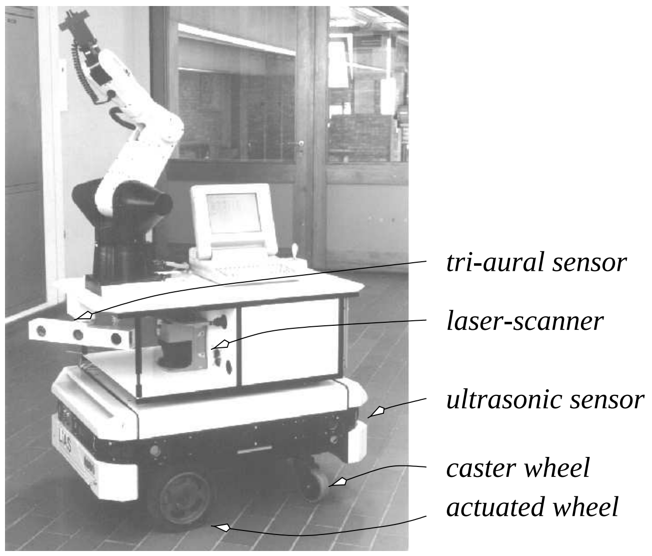


Figure 9.1: The "*LiAS*" mobile robot of K.U.Leuven-PMA with two caster wheels and two driven wheels. The robot is equipped with (i) an array ultrasonic sensors around the perimeter (in the black stripe just above the wheels); (ii) an independently moving set of three ultrasonic sensors (*tri-aural* sensor, [16]); (iii) a laser scanner (not well visible at the centre of the vehicle); and (iv) gyroscopes (hidden). (Photograph courtesy of J. Vandorpe.)

Applications. Mobile robots are used for different purposes than serial and parallel manipulators, i.e., they mainly transport material or tools over distances much larger than their own dimensions. They have to work in environments that are often cluttered with lots of known and unknown, moving and immobile obstacles. The requirements on their absolute and relative accuracies, as well as on their operation speeds, are about an order of magnitude less stringent, i.e., of the order of one centimeter and ten centimeters per second, respectively. The last decade, much effort has been spent in automation of truck and car navigation, in constrained areas such as container harbours, or more "open" areas such as highways.

Special kinematic designs. Most academic or industrial mobile robots have different kinematic features with respect to real-world trucks or passenger cars, for the sole reason of *kinematic simplicity.* The major simplifications are

1. *Two independently driven wheels.* This allows accurate measurement and control of the wheels' rotation. The speed difference between both wheels generates rotation of the vehicle. Moreover, vehicles equipped with two independently driven wheels can rotate on the spot.

2. *No suspensions.* In normal cars, the suspension compensates discomforting influences of the car's dynamics at high speeds and high disturbances. This goal is achieved by deformation and/or relative displacement of some parts in the suspension. This means that the position and orientation of the wheels cannot be *measured* and *controlled* directly. For this reason, mobile robots don't have suspensions. Hence, their speeds are limited.

3. *Holonomic* mobile robots (also called *omnidirectional* vehicles) have been developed in many academic and industrial research labs. These devices use special types of wheels or wheel-like artifacts as in Fig. 9.2, [1, 12], or spheres driven by three or more rollers, [8, 19, 20].

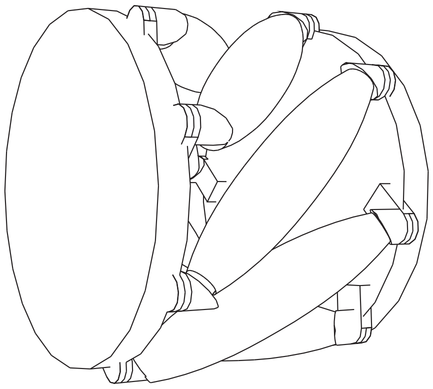


Figure 9.2: Example of a holonomic wheel. The passive "rollers" allow rolling of the wheel in all directions.

Fact-to-Remember 60 (Basic ideas of this Chapter)
Mobile robots are simpler than serial and parallel robots since only planar motions are involved. They are more complex than serial and parallel robots, because of the nonholonomic constraints on their instantaneous velocity. At the velocity level, mobile robots behave as a special type of parallel robot (i.e., it has different connections to the ground), such that this Chapter requires no new concepts.

9.2 Rigid body pose in SE(2)

A mobile robot is a rigid body in E^3 which is constrained to move in a plane. Whenever coordinate representations are used, this text assumes that the XY-planes of (orthogonal) reference frames coincide with this plane. The robots' *position and orientation* are given by three parameters with respect to a world or base reference frame $\{bs\}$, Fig. 9.3: the X and Y components (x, y) of the origin of the "end-effector" reference frame on the robot and the angle ϕ between the X-axes of base and end-effector frames. The end-effector frame is usually chosen to coincide with the midpoint of the actuated wheel axle, for several reasons: the nonholonomic constraint gets a simple coordinate expressions, and the influence of left and right driven wheels are symmetric. However, the kinematics presented in the following Sections are independent of the choice of reference frames.

Since SE(2) (the configuration space of mobile robots) is a sub-group of SE(3) (the configuration space of all rigid body poses), all concepts introduced in the previous Chapters are re-used in this Chapter in the simplified form described in the following paragraphs.

Pose. The homogeneous transformation matrix (Eq. (6.1) in Sect. 6.2) reduces to (Fig. 9.3):

$$ {}^b_a\boldsymbol{T} \triangleq \begin{pmatrix} & x \\ {}^b_a\boldsymbol{R}(Z,\phi) & y \\ & 0 \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix}, \quad \text{with} \quad {}^b_a\boldsymbol{R}(Z,\phi) = \begin{pmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{9.1} $$

In the context of mobile robot kinematics, this is simplified to either a 3×3 matrix (denoted by the same symbol ${}^{b}_{a}\boldsymbol{T}$) or a finite displacement three-vector twist $\mathbf{t}_d$:

$$
{}^{b}_{a}\boldsymbol{T} = \begin{pmatrix} \cos(\phi) & -\sin(\phi) & x \\ \sin(\phi) & \cos(\phi) & y \\ 0 & 0 & 1 \end{pmatrix}, \qquad \mathbf{t}_d = \begin{pmatrix} \phi \\ x \\ y \end{pmatrix}. \tag{9.2}
$$

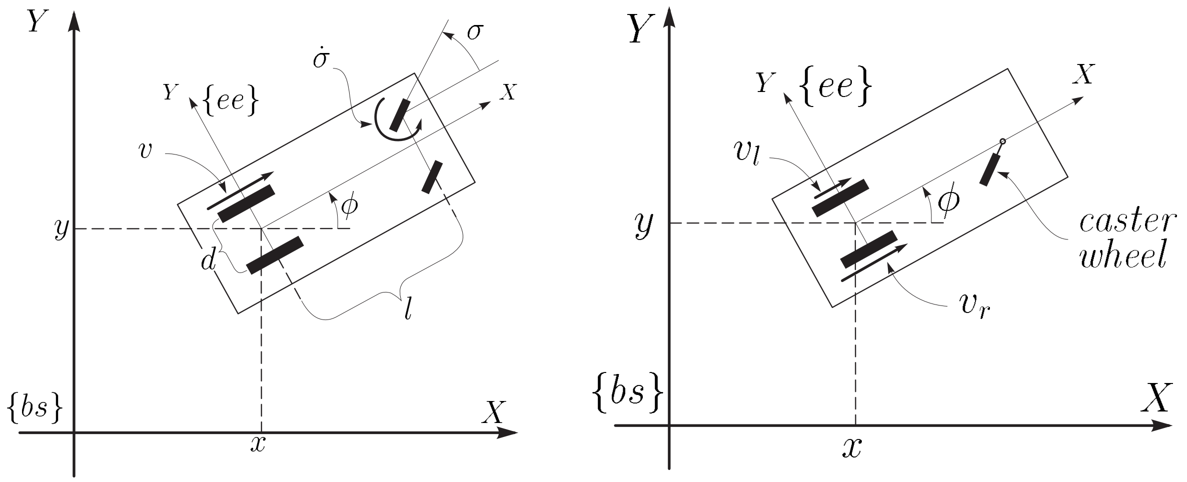


Figure 9.3: Geometric parameters of car-like robot (left) and differentially-driven robot (right).

Screw. The instantaneous screw axis (ISA) of Chasles' Theorem (Fact. 13) reduces to an *instantaneous centre of (pure) rotation* (ICR): the ISA is always *orthogonal* to the plane of the motion, and the ICR lies at the intersection of this plane and the ISA. Similarly, Poinsot's Theorem reduces to an *instantaneous line of (pure) force* (ILF) *in* the plane. The coordinate six-vector of a mobile robot twist always contains three zeros, hence it is represented by a three-vector (denoted by the same symbol):

$$
\mathbf{t}_{SE(3)} = \begin{pmatrix} 0 \\ 0 \\ \omega \\ v_x \\ v_y \\ 0 \end{pmatrix} \Rightarrow \mathbf{t}_{SE(2)} = \begin{pmatrix} \omega \\ v_x \\ v_y \end{pmatrix}. \tag{9.3}
$$

Similarly, the 2D wrench three-vector becomes $\mathbf{w} = (f_x \; f_y \; m)^T$, but it results from putting to zero three *other* elements in the 3D screw:

$$
\mathbf{w}_{SE(3)} = \begin{pmatrix} f_x \\ f_y \\ 0 \\ 0 \\ 0 \\ m \end{pmatrix} \Rightarrow \mathbf{w}_{SE(2)} = \begin{pmatrix} f_x \\ f_y \\ m \end{pmatrix}. \tag{9.4}
$$

A screw twist transforms with the 3D screw transformation matrix simplified as follows:

$$\begin{pmatrix} {}_a\omega \\ {}_av_x \\ {}_av_y \end{pmatrix} = {}^b_a\boldsymbol{S}_{\mathbf{t}} \begin{pmatrix} {}_b\omega \\ {}_bv_x \\ {}_bv_y \end{pmatrix} \quad \text{with} \quad {}^b_a\boldsymbol{S}_{\mathbf{t}} = \begin{pmatrix} 1 & 0 & 0 \\ y & c_\phi & -s_\phi \\ -x & s_\phi & c_\phi \end{pmatrix}. \tag{9.5}$$

Wrenches transform with the 3D screw transformation matrix simplified as follows:

$$\begin{pmatrix} {}_af_x \\ {}_af_y \\ {}_am \end{pmatrix} = {}^b_a\boldsymbol{S}_{\mathbf{w}} \begin{pmatrix} {}_bf_x \\ {}_bf_y \\ {}_bm \end{pmatrix} \quad \text{with} \quad {}^b_a\boldsymbol{S}_{\mathbf{w}} = \begin{pmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ -yc_\phi + xs_\phi & ys_\phi + xc_\phi & 0 \end{pmatrix} \begin{pmatrix} \omega \\ {}_bv_x \\ {}_bv_y \end{pmatrix}. \tag{9.6}$$

The reciprocity between twists and wrenches, Eq. (3.7), becomes:

$$\mathbf{t}^T \widetilde{\boldsymbol{\Delta}} \mathbf{w} = 0, \quad \text{with} \quad \widetilde{\boldsymbol{\Delta}} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad \text{and} \quad \widetilde{\boldsymbol{\Delta}}^{-1} = \widetilde{\boldsymbol{\Delta}}^T. \tag{9.7}$$

9.3 Kinematic models

9.3.1 Equivalent robot models

Real-world implementations of car-like or differentially-driven mobile robots have three or four wheels, because the robot needs at least three non-collinear support points in order not to fall over. However, the kinematics of the moving robots can be described by simpler *equivalent* robot models: a "*bicycle*" robot for the car-like mobile robot (i.e., the two driven wheels are replaced by one wheel at the midpoint of their axle, whose velocity is the mean v_m of the velocities v_l and v_r of the two real wheels) and a "caster-less" robot for the differentially-driven robot (the caster wheel has no kinematic function; its only purpose is to keep the robot in balance). In addition, Fig. 9.4 shows how car-like and differentially-driven mobile robots can be modelled by an *equivalent (planar) parallel robot*, consisting of three RPR-legs (passive revolute joint, actuated prismatic joint, passive revolute joint). The nonholonomic constraint is represented by a zero actuated joint velocity v_c in the leg on the wheel axles. A car-like robot has two such constraints; a differentially-driven robot has one. Since the constraint is nonholonomic and hence not integrable, the equivalent parallel robot is only an *instantaneous* model, i.e., the base of the robots moves together with the robots. Hence, the model is only useful for the velocity kinematics of the mobile robots. The velocities in the two kinematic chains on the rear wheels of the car-like robot are not independent; in the rest of this Chapter they are replaced by one single similar chain connected to the midpoint of the rear axle (shown in dashed line in Fig. 9.4).

The car-like robot model in Figure 9.5 is only an approximation, because neither of the two wheels has an orientation that corresponds exactly to the steering angle σ. In fact, in order to be perfectly outlined, a steering suspension should orient both wheels in such a way that their perpendiculars intersect the perpendicular of the rear axle in the same point. In practice, this is never perfectly achieved, so one hardly uses car-like mobile robots when accurate motion is desired. Moreover, the two wheels of a real car are driven through a *differential gear transmission*, in order to divide the torques over both wheels in such a way that neither of them slips. As a result, the mean velocity of both wheels is the velocity of the drive shaft.

9.3.2 Centre of rotation

Figure 9.5 shows how the instantaneous centre of rotation is derived from the robot's pose (in the case of a car-like mobile robot) or wheel velocities (in the case of a differentially-driven robot). The *magnitude* of the

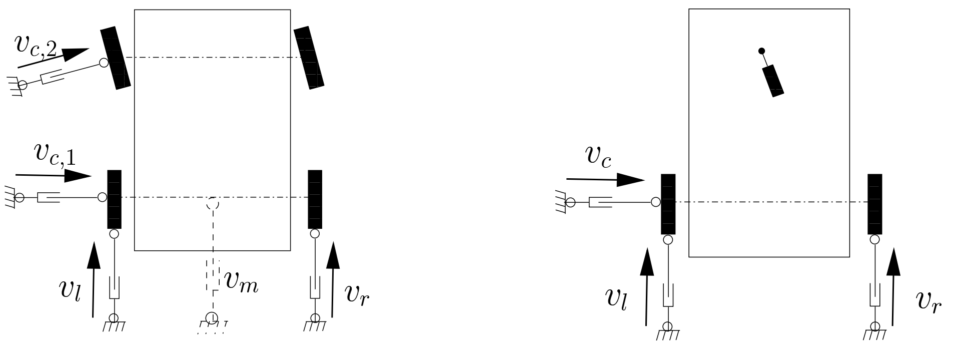


Figure 9.4: Instantaneously equivalent parallel manipulator models for car-like robot (left) and differentially-driven robot (right).

instantaneous rotation is in both cases determined by the magnitudes of the wheel speeds; the distance between the instantaneous centre of rotation and the wheel centre points is called the *steer radius*, [18], or *instantaneous rotation radius* r^{ir}. Figure 9.5 and some simple trigonometry show that

$$r^{ir} = \begin{cases} \dfrac{l}{\tan(\sigma)}, & \text{for a car-like robot,} \\ \dfrac{d}{2}\dfrac{v_r + v_l}{v_r - v_l}, & \text{for a differentially-driven robot.} \end{cases} \tag{9.8}$$

with l the *wheelbase*, [18]), (i.e., the distance between the points where both wheels contact the ground), σ the steer angle, d the distance between the wheels of the differentially-driven robot, and v_r and v_l its wheel velocities (Fig. 9.3).

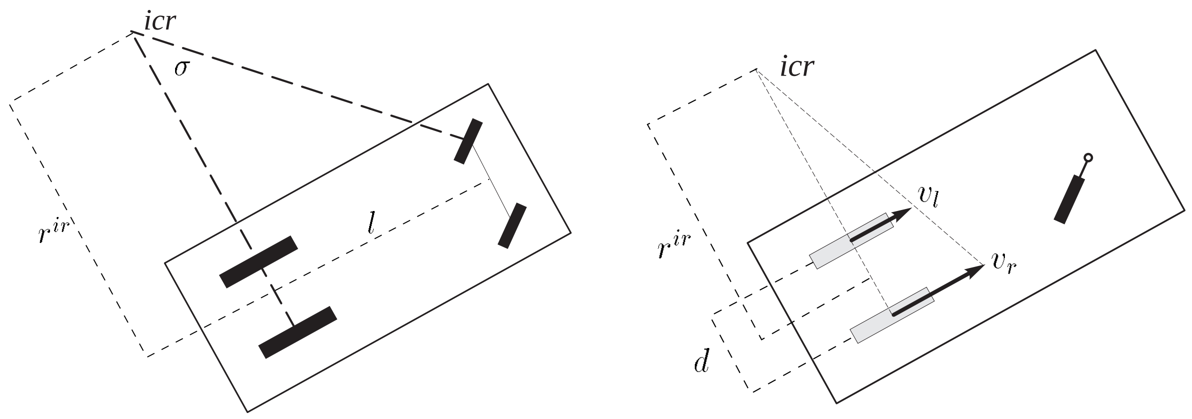


Figure 9.5: Instantaneous centre of rotation (*icr*) for car-like (left) and differentially-driven robots (right).

Fact-to-Remember 61 (Car-like vs. differentially-driven robots)
Differentially-driven robots have two instantaneous degrees of motion freedom, compared to one for car-like robots. A car-like mobile robot must drive forward or backwards if it wants to turn but a differentially-driven robot can turn on the spot by giving opposite speeds to both wheels. The instantaneous rotation centre of differentially-driven robots can be calculated more accurately than that of car-like robots, due to the absence of two steered wheels with deformable suspensions.

9.3.3 Mobile robot with trailer

Trailers can be attached to a mobile robot, in order to increase the load capacity of the system. The instantaneous rotation centre for the trailer depends on the *hinge angle* α between truck and trailer, as well as on the *hookup length* l_h between the axle of the trailer and the attachment point on the axle of the truck (Fig. 9.6). The kinematics become slightly more complicated if the trailer is not hooked up *on* the rear axle of the truck. An interesting special case are the luggage carts on airports: the trailers follow the trajectory of the pulling truck (more or less) *exactly*. It can be proven that this behaviour results from using a hinge exactly in the middle between the axles of tractor and trailor, [7]. In general, the hinge is not in the middle, but closer to the truck axle; the truck-and-trailer system then needs a wider area to turn than the truck alone.

Finally, note that the system truck-and-trailer becomes even more constrained than the single truck alone: the two actuated degrees of freedom remain (i.e., one first-order and one second-order), but they now have to drive *six* Cartesian degrees of freedom, three of the truck and three of the trailer.

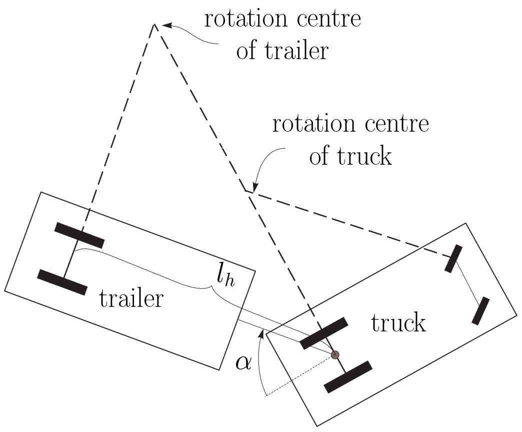


Figure 9.6: Instantaneous rotation centres of truck with trailer.

9.4 Forward force kinematics

Mobile robots have instantaneously equivalent parallel robot models. Hence, as in the case of parallel robots, the forward force kinematics are the easiest mapping between joint space and end-effector space. The FFK (Sect. 8.6) uses the wrench basis $\boldsymbol{G}$, Eq. (8.8), of the equivalent parallel manipulator. Since differentially-driven robots and car-like robots have different parallel manipulator models, their wrench bases are different too:

$$\mathbf{w}^{ee} = {}_{cast}\boldsymbol{G} \begin{pmatrix} f_r \\ f_l \\ f_c \end{pmatrix}, \qquad \mathbf{w}^{ee} = {}_{car}\boldsymbol{G} \begin{pmatrix} f_m \\ f_{c,1} \\ f_{c,2} \end{pmatrix}. \tag{9.9}$$

f_r and f_l are the traction forces required in the left and right wheel of the differentially-driven robot to keep the end-effector wrench $\mathbf{w}^{ee}$ in static equilibrium; similarly, f_m is the sum of the traction forces on both wheels of the car-like robot. The $f_{c,\cdot}$ are the forces generated by $\mathbf{w}^{ee}$ in the constraint directions. The wrench basis $\boldsymbol{G}$ consists of the *partial wrench* of the actuated prismatic joints, Sect 7.12.1. These are easily derived by inspection of Fig. 9.4: they are pure forces on the end-effector of the leg and through the axes of the two revolute joints. Hence, for a differentially-driven robot the coordinate expression of $\boldsymbol{G}$ with respect to the end-effector frame $\{ee\}$ at the midpoint of the wheel axle (Figs 9.3) is:

$${}_{ee}\boldsymbol{G}_{dd} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ \frac{d}{2} & -\frac{d}{2} & 0 \end{pmatrix}, \tag{9.10}$$

with d the distance between both wheels. For a car-like robot, $\boldsymbol{G}$ is most easily expressed in a frame $\{icr\}$ (parallel to $\{ee\}$ and with origin at the instantaneous centre of rotation), since both constraint manipulator legs intersect at that point. The coordinate expression of $\boldsymbol{G}$ in $\{icr\}$ is:

$${}_{icr}\boldsymbol{G}_{car} = \begin{pmatrix} 1 & 0 & \cos(\sigma) \\ 0 & -1 & -\sin(\sigma) \\ r^{ir} & 0 & 0 \end{pmatrix}, \tag{9.11}$$

with r^{ir} the distance to the instantaneous centre of rotation, Eq. (9.8). Pre-multiplication with the screw transformation matrix ${}^{icr}_{ee}\boldsymbol{S}_{\mathbf{w}}$ gives the expression in the end-effector frame $\{ee\}$:

$${}_{ee}\boldsymbol{G}_{car} = \begin{pmatrix} 1 & 0 & c_\sigma \\ 0 & -1 & -s_\sigma \\ -r^{ir} & 0 & 1 \end{pmatrix} {}_{icr}\boldsymbol{G}_{car} = \begin{pmatrix} 1 & 0 & c_\sigma \\ 0 & -1 & -s_\sigma \\ 0 & 0 & -r^{ir}c_\sigma \end{pmatrix}. \tag{9.12}$$

9.5 Inverse velocity kinematics

Again using the equivalent parallel robot, the IVK for the example of the differentially-driven robot corresponds to:

$$\begin{pmatrix} v_r \\ v_l \\ v_c \end{pmatrix} = (\widetilde{\boldsymbol{\Delta}}\boldsymbol{G}_{dd})^T \mathbf{t}^{ee}, \quad \text{and} \quad \begin{pmatrix} v_m \\ v_{c,1} \\ v_{c,2} \end{pmatrix} = (\widetilde{\boldsymbol{\Delta}}\boldsymbol{G}_{car})^T \mathbf{t}^{ee}. \tag{9.13}$$

v_r and v_l are the velocities of the right and left wheels (Fig. 9.4); v_m is the velocity of the differential on the rear wheel axle; $v_c, v_{c,1}$ and $v_{c,2}$ are the velocities in the constrained directions. The nonholonomicity constraint corresponds to:

$$v_c = v_{c,1} = v_{c,2} = 0. \tag{9.14}$$

The IVK has two important applications:

1. The *desired* end-effector twist $\mathbf{t}^{ee}$ that is generated in motion planning and/or control software for mobile robots must be such that these constraints are satisfied. If this is not the case, one could apply the *kinetostatic filtering* of Sect. 7.15.2 to project the nominally desired end-effector twist $\mathbf{t}^{ee}$ onto the twist space of reachable velocities. This projection involves a weighting between translational and rotational velocities. During motion control of the mobile robot, this means that a choice has to be made between reducing errors in x or y ("distance"), or ϕ ("heading") independently. For example, no linear control law can achieve reduction in a transversal error (i.e., Y in the $\{ee\}$) if the errors in either X or ϕ are zero, [11].

2. External sensors can produce *measured* end-effector twists $\mathbf{t}^{ee}$. Using the IVK in Eq. (9.13) then yields the corresponding wheel velocities *and* the transversal slip velocity.

9.6 Forward velocity kinematics

The forward velocity kinematics (FVK) for mobile robots tackles the following problems:

1. Differentially-driven robots: *"If the motor velocities $\dot{q}_r$ and $\dot{q}_l$ of the right and left wheels are known, as well as the wheel radii r_r and r_l, the distance d between both wheels, and the current pose $(\phi\, x\, y)^T$ of the robot, what is then its corresponding twist $\mathbf{t}^{ee}$?"*

2. Car-like robots: *"If the drive shaft rotation velocity $\dot{q}$ and the steering angle σ are known, as well as the radius r of the equivalent wheel, the wheelbase l, and the current pose $(\phi\, x\, y)^T$ of the robot, what is then its corresponding twist $\mathbf{t}^{ee}$?"*

The wheel and motor velocities are linked, e.g., $v_l = r_l \dot{q}_l$.

Assumptions. Since the contact between wheels and floor relies on *friction*, the accuracy of the FVK heavily depends on how well the following constraints are satisfied:

1. *Non slipping.* The wheels do not slip *transversally*, i.e., they obey the nonholonomicity constraint.

2. *Pure rolling.* The wheels do not slip *longitudinally*, so that the distance over which the outer wheel surface rotates equals the distance travelled by the point on the rigid body to which the wheel axle is attached.

3. *Constant wheelbase.* The wheels constantly contact the ground in different points, due to the combined influence of elasticity of the wheels and non-planarity of the ground.

4. *Constant wheel diameter.* Most wheels have non-negligible compliance, such that (dynamical) changes in the load on each wheel generate changes in the wheel diameter. Hence, the relationship between motor and wheel velocities changes too.

In order to prevent violation of these assumptions, the mobile robot should (at least) avoid *jumps* in its motion since this is a major cause of slippage. The Chapter on motion planning will present some trajectories that have continuous *jerk*, i.e., the acceleration of the mobile robot has no sudden jumps. In terms of your car driving experience this translates into the property that one doesn't have to turn the steering wheel abruptly.

FVK algorithm. The FVK of the mobile robots follow straightforwardly from the FVK of the equivalent parallel robots, Eq. (8.22). For example, in the differentially-driven robot case:

$$\mathbf{t}^{ee} = (\widetilde{\mathbf{\Delta}} \boldsymbol{G})^{-T} \begin{pmatrix} v_r \\ v_l \\ v_c = 0 \end{pmatrix}, \tag{9.15}$$

and a similar expression for the car-like robot. The inverses of the wrench bases in Eqs (9.10) and (9.12 are easily found:

$$(\widetilde{\mathbf{\Delta}}\ \boldsymbol{G}_{dd})^{-1} = \begin{pmatrix} \frac{1}{2d} & \frac{1}{2} & 0 \\ -\frac{1}{2d} & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \text{and} \quad (\widetilde{\mathbf{\Delta}}\ \boldsymbol{G}_{car})^{-1} = \begin{pmatrix} \frac{1}{r^{ir}} & 1 & 0 \\ \frac{\tan(\phi)}{r^{ir}} & 0 & -1 \\ -\frac{1}{r^{ir}\cos(\phi)} & 0 & 0 \end{pmatrix}. \tag{9.16}$$

So the FVK with respect to the midframe on the rear axle are given by:

$$\mathbf{t}^{ee}_{dd} = \begin{pmatrix} \omega \\ v_x \\ v_y \end{pmatrix} = \begin{pmatrix} \frac{v_r - v_l}{2d} \\ \frac{v_r + v_l}{2} \\ 0 \end{pmatrix}, \quad \text{and} \quad \mathbf{t}^{ee}_{car} = \begin{pmatrix} \frac{v_m}{r^{ir}} \\ v_m \\ 0 \end{pmatrix}. \tag{9.17}$$

These FVK relationships could of course also be derived by inspection.

Fact-to-Remember 62 (Velocity kinematics for mobile robots)
The forward velocity kinematics are straightforward, but subject to inaccurate modelling approximations. The inverse velocity kinematics are, in general, not uniquely defined.

9.7 Forward position kinematics

9.7.1 Dead reckoning—Odometry

The forward position kinematics of a mobile robot (i.e., estimating its pose from wheel encoder sensing only) is called *dead reckoning* (a term originating in ship and airplane navigation) or *odometry* (from the Greek words "hodos" and "metron", meaning "road" and "measure" respectively), [18]. (An interesting historical note: the Chinese invented a mechanical *hodometer* already in about 265 AD!) Contrary to the trivial case of (holonomic) serial and parallel robots, dead reckoning for (nonholonomic) mobile robots *must* be performed by integration of velocity equations such as Eq. (9.17). The literature contains several numerical integration procedures, such as Euler's scheme, the trapezoidal rule, or the family of Runge-Kutta rules, see e.g., [9]. Recall that general rigid body velocities are not integrable, Sect. 5.3.6, due to the non-integrability of the angular velocity. For motions on a plane, however, the angular velocity *is* integrable. This does not necessarily imply the integrability of the total twist of a mobile robot: a *pose twist* is integrable (or "exact"), but a *screw* twist is not. Moreover, any integration scheme will experience *drift*, due to numerical round-off errors. In the case of mobile robots, however, this drift is increased by (i) the finite resolution of the sensors, (ii) the inaccuracies in the geometric model (deformations), and (iii) slippage. Hence, the robot needs extra sensors to recalibrate regularly its pose with respect to its environment.

9.7.2 Sensors for mobile robots

By nature of both their nonholonomic kinematic character and the kind of environments in which they operate, mobile robots are often equipped with both *proprioceptive* and *exteroceptive* sensors, [2]:

1. *Proprioceptive sensors* measure the "internal" state parameters of the robot: the motor positions and/or speeds of the driving wheels and the steering wheel. The hardware used for these measurements is not different from the revolute joint angle and velocity sensors used in serial manipulators, i.e., encoders and/or resolvers, [3].

2. *Exteroceptive sensors* measure "external" motion parameters of the robot, such as

   (a) Its *absolute orientation*, by means of, for example, a gyroscope.

   (b) Its *absolute position*, by means of, for example, a (D)GPS ((Differential) Global Positioning System), which triangulates between signals from different satellites. The standard systems' *absolute accuracy* of 20–100m is such that this option is only useful during long-distance trajectories, not for motions within one single building. Applications exist already in, for example, automatic harvesting. For indoor applications, more accurate systems working with artificial "satellites" placed on the room's ceiling are being used more and more often.

   (c) The orientation of (known or unknown, expected or unexpected) *landmarks* in the robot's environment, by means of, for example, cameras or a laser scanner that detects "bright spots" or digital code strips on the landmarks.

   (d) The *distance* to objects in the environment, by means of ultrasonic sensors, or, more accurately, laser range finders.

   From these measurements, and from some triangulation, the robot can estimate its relative pose with respect to the landmarks. These external sensors serve three complementary purposes: (i) position and orientation estimation; (ii) environment map building; and (iii) obstacle avoidance. (The Chapter on intelligent sensor processing will discuss the latter two goals in more detail.) Their pose *estimation* capabilities are much worse than the pose *sensing* capabilities of serial and parallel manipulators.

9.8 Inverse position kinematics

In principle, the inverse position kinematics for a mobile robot would have to solve the following problem: *Given a desired pose* $(\phi\, x\, y)^T$ *for the robot, what are the wheel joint angles that would bring the robot from its current pose to the desired pose?"* However, this problem is much more complicated than the corresponding problem for serial and parallel manipulators, for several reasons: (i) infinitely many possible solutions exist; (ii) no analytical (or, closed-form) decomposition approach is known, as was the case for serial and parallel robots; (iii) classical linear control theory is not sufficient, [4, 5, 15, 17]. Some solution techniques will be presented in the Chapter on motion planning.

9.9 Motion operators

From the previous Sections, it should be clear that a mobile robot cannot move instantaneously in all directions, but that it is capable to reach all possible poses in a plane. This last capacity requires some *motion planning* that can be rather involved. The Chapter on motion planning will discuss these aspects in more detail, but here we can already define some basic *motion operators* for mobile robots, [6, 14].

For differentially-driven robots, these two basic operators are:

ROTATE By applying *opposite* velocities to both wheels, a differentially-driven robot rotates instantaneously about the midpoint of its wheel axle.

DRIVE By applying *equal* velocities to both wheels, a differentially-driven robot moves along its longitudinal axis.

The third "motion degree of freedom" (i.e., moving along the direction of the wheel axis) is approximated by the **SLIDE** operator, that is the *commutator* (or Lie bracket) of ROTATE and DRIVE: SLIDE = $\mathrm{R}^{-1}\mathrm{D}^{-1}\mathrm{RD}$, with D denoting the DRIVE operation, and R the ROTATE operation. The SLIDE operator must be interpreted as follows: by driving a "little bit" forwards (D), then rotating a little bit to the left (R), then driving a little bit backwards (D^{-1}), and finally rotating a little bit to the right (R^{-1}), the robot ends up in a position that is translated a little bit along its wheel axle direction. The SLIDE operator becomes a transversal *velocity* in the limit case that "little bit" becomes zero, i.e, (i) the travelled distances go to zero, and (ii) the motion times for each operator go to zero too. Of course, this limit cannot be attained in practice!

For car-like robots, the two basic operators are:

DRIVE By applying a velocity to the wheel axle, a car-like robot moves along its longitudinal axis, or rather, it rotates about the instantaneous rotation centre determined by the steering angle.

STEER By applying an angular velocity to the steering wheel, the direction of the motion generated by the driving wheels of a car-like robot can be changed. Unlike all previously defined operators, the STEER operator for a car-like robot does *not* induce a Cartesian motion, but only a change in the *state* of the robot.

The commutator of STEER and DRIVE generates a **ROTATE** operator, that rotates the robot about the midpoint of its wheel axle. As in the case of the differentially-driven robot, the commutator of this ROTATE operator and the DRIVE operator generates the **SLIDE** operator.

Most of the motion operators defined above correspond to a particular choice of *basis twists* in the instantaneous twist space of the mobile robot. However, the STEER operator for car-like robots is *not* a twist: it is a second-order operator that generates no velocity by itself.

References for this Chapter

[1] J. Agulló, S. Cardona, and J. Vivancos. Kinematics of vehicles with directional sliding wheels. *Mechanism and Machine Theory*, 22(4):295–301, 1987.

[2] B. Barshan and H. F. Durrant-Whyte. Inertial navigation systems for mobile robots. *IEEE Trans. Rob. Automation*, 11(3):328–342, 1995.

[3] J. P. Bentley. *Principles of measurement systems*. Longman Scientific & Technical, 1995.

[4] R. W. Brockett. Nonlinear systems and nonlinear estimation theory. In M. Hazewinkel and J. C. Willems, editors, *Stochastic systems : the mathematics of filtering and identification and applications*, pages 441–477, Reidel, 1981.

[5] R. W. Brockett. Control theory and singular riemannian geometry. In *New Directions in Applied Mathematics*, pages 11–27. Springer, 1982.

[6] W. L. Burke. *Applied differential geometry*. Cambridge University Press, 1992.

[7] L. G. Bushnell, B. Mirtich, A. Sahai, and M. Secor. Off-tracking bounds for a car pulling trailers with kingpin hitching. In *33rd IEEE Conf. on Decision and Control*, pages 2944–2949, 1994.

[8] L. Ferrière, B. Raucent, and G. Campion. Design of omnimobile robot wheels. In *IEEE Int. Conf. Robotics and Automation*, pages 3664–3670, 1996.

[9] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Addison-Wesley, 1990.

[10] H. Goldstein. *Classical mechanics*. Addison-Wesley, 1980.

[11] Z.-P. Jiang and H. Nijmeijer. Tracking control of mobile robots: A case study in backstepping. *Automatica*, 33(7):1393–1399, 1997.

[12] P. F. Muir and C. P. Neuman. Resolved motion rate and resolved acceleration servo-control of wheeled mobile robots. In *IEEE Int. Conf. Robotics and Automation*, pages 1133–1140, 1990.

[13] J. I. Neimark and N. A. Fufaev. *Dynamics of nonholonomic systems*. American Mathematical Society, Providence, 1972.

[14] E. Nelson. *Tensor analysis*. Princeton University Press, Princeton, 1967.

[15] S. Patarinski, H. Van Brussel, and H. Thielemans. Kinematics and control of wheeled mobile robots. In *Intelligent Robots*, pages 177–186, 1993.

[16] H. Peremans, K. Audenaert, and J. M. Van Campenhout. A high-resolution sensor based on tri-aural perception. *IEEE Trans. Rob. Automation*, 9(1):36–48, 1993.

[17] C. Samson and K. Ait-Abderrahim. Feedback control of a nonholonomic wheeled cart in Cartesian space. In *IEEE Int. Conf. Robotics and Automation*, pages 1136–1141, 1991

[18] B. Steer. Trajectory planning for a mobile robot. *Int. J. Robotics Research*, 8(5):3–14, 1989.

[19] M. Wada and S. Mori. Holonomic and omnidirectional vehicle with conventional tires. In *IEEE Int. Conf. Robotics and Automation*, pages 3671–3676, 1996.

[20] H. West and H. Asada. Design of ball wheel mechanisms for omnidirectional vehicles with full mobility and invariant kinematics. *Trans. ASME J. Mech. Design*, 119:153–161, 1997.

Chapter 10

Dynamics

10.1 Introduction

Dynamics studies how the forces acting on bodies make these bodies move. This text is limited to *rigid body* dynamics, starting from Newton's laws describing the dynamics of a *point mass*. Most of the introductory material can be found in textbooks on classical physics and mechanics, e.g., [1, 5, 12, 13, 28, 34]. Research on the dynamics of *multiple rigid bodies* has most often not been performed with only humanoid robots in mind, but has progressed (more or less independently) in different research communities, each with their own emphasis:

1. *Serial robot arms*, [21, 30], with the emphasis on efficient, real-time algorithms. "Real time" means that the robot control computer must be able to perform these calculations about 1000 times per second. A lot of attention has also gone into *redundancy resolution*, *singularity avoidance*, and *(on-line) parameter identification.* The human arm is highly redundant, and hence very dextrous and versatile. But it can end up in singular configurations too: when, for example, your arm and wrist are completely stretched out, you cannot move any further in the direction along the arm.

   All dynamics algorithms discussed in this text assume that the physical parameters of the robot are rather accurately known: dimensions of links, relative positions and orientations of connected parts, mass distribution of links, joints and motors. Hence, advanced parameter identification techniques are required for humanoid robots, because these type of robots will most likely have to carry or push unknown loads.

2. *Computer graphics and animation.* Here, the emphasis is on *realistically looking* interactions between different (rigid and soft) bodies. Most often, bodies are not actuated by motors, but are falling onto each other, hit by projectiles, or they are racing cars that have only one motorized degree of freedom. Computer graphics is growing closer and closer to robotics, by paying more attention to the realistic simulation of human figures.

3. *Spacecraft control*, mainly investigating the effects of a free-floating basis, and structural flexibilities in moving parts such as solar panels, [17, 18].

4. *Modelling of cars, trucks and trains*, taking into account a large number of bodies and motion constraints, non-linear elements (such as real springs, dampers, and friction), and putting much emphasis on the development of numerical integration schemes than can cope with the system's large dimensions.

Humanoid robots require the integration of results from all above-mentioned research areas:

- *Serial substructures.* Basically, a humanoid robot consists of a number of serial parts: legs, arms, and head, all connected to the same trunk, which in itself might consist of several bodies connected in series.

- *Real-time control.* Humanoid robots must be able to calculate their motor torques in real time, otherwise walking or running on two legs is impossible. The computational complexity of an algorithm is most often expressed as $\mathcal{O}(N^k)$ ("order N to the kth power"), where N is the number of rigid bodies in the robot. $\mathcal{O}(N^k)$ means that the time to compute the dynamics increases proportionally to the kth power of the number of bodies in the system. This text deals only with the most computationally efficient case, where $k = 1$, i.e., so-called *linear-time* algorithms.
- *Motion constraints.* The feet of the humanoid robot are in contact with the ground; the arms can grasp objects that are fixed in the environment; contacts with the environment result in closed kinematic loops (e.g., with two feet on the ground, the motions of both legs are not independent because the ground acts as a rigid link between both feet). All these interactions constrain the relationships between motor torques and resultant motion to lie on lower-dimensional "constraint manifolds."
- *Free-floating base.* When running, both feet are in the air, and there is no fixed support point.
- *Redundancy resolution.* Humanoid robots have a lot of redundancy, such that the same task can be executed in infinitely many ways. This allows for *optimization* of the task, as well as for performing *lower-priority subtasks* together with the main task. For example: avoidance of an obstacle or of a singular configuration by the legs, trunk and arms, while the hands transport the load along the desired trajectory.

10.2 Forward and Inverse Dynamics

The *Forward Dynamics* (FD) algorithm solves the following problem: *"Given the vectors of joint positions $\boldsymbol{q}$, joint velocities $\dot{\boldsymbol{q}}$, and joint forces $\boldsymbol{\tau}$, as well as the mass distribution of each link, find the resulting end-effector acceleration $\ddot{\boldsymbol{X}}$."* (We use the notation "$\ddot{\boldsymbol{X}}$" for the six-dimensional coordinate vector of a rigid body acceleration, although, strictly speaking, it is *not* the second-order time derivative of any six-dimensional representation of position and orientation; also the velocity $\dot{\boldsymbol{X}}$ is not such a time derivative. However, this notation is often used in the dynamics literature.) The FD are used for *simulation* purposes: find out what the robot does when known joint torques are applied.

Similarly, the *Inverse Dynamics* (ID) algorithm solves the following problem: *"Given the vectors of joint positions $\boldsymbol{q}$, joint velocities $\dot{\boldsymbol{q}}$, and desired joint accelerations $\ddot{\boldsymbol{q}}$, (or end-effector acceleration $\ddot{\boldsymbol{X}}$) as well as the mass matrix of each link, find the vector of joint forces $\boldsymbol{\tau}$ required to generate the desired acceleration."* The ID are needed for:

1. *Control*: if one wants the robot to follow a specified trajectory, one has to convert the desired motion into the joint forces that will generate this motion.
2. *Motion planning*: when generating a desired motion for the robot end-effector, one can use the ID of the robot to check whether the robot's actuators will be able to generate the joint forces needed to execute the trajectory.

Finding algorithms to calculate the dynamics is much more important for *serial* robots than for parallel or mobile robots: the dynamics of parallel and mobile robots are reasonably approximated by the dynamics of *one single* rigid body. Moreover, mobile robots move so slowly (in order to avoid slippage) and their inertia changes so little that dynamic effects are small. Parallel robots, on the other hand, have light links, and all motors are in, or close to, the base, such that the contributions of the manipulator inertias themselves are limited. Hence, this Chapter treats the dynamics of serial manipulators only; the interested reader is referred to the literature (e.g., [3, 4, 19, 24, 26, 27, 33]) for more details on the dynamics of mobile and parallel robots.

This text takes into account the dynamics of the robot links only, *not* that of the motors. Just be aware that the motor inertia can be very significant, *especially* for the high gear ratios between motor shaft and robot link shaft as used in most industrial robots.

Parametric uncertainty. The FD and ID algorithms in this Chapter use *models*: kinematic models (i.e., the relative positions and orientations, and relative velocities of all joints), and dynamic models (i.e., the mass distribution of each link). Of course, in real-world systems, this information is seldom known with high accuracy, such that *calibration* or *identification* of the kinematic and dynamic parameters is required whenever high absolute static and dynamic accuracy are desired. This Chapter treats the dynamics of *ideal* kinematic structures only, i.e., they exhibit no friction, no backlash, and no flexibility. In general, these non-ideal effects increase when transmissions between motors and joint axes are used.

10.3 Tree-structure topology

All joints in a typical humanoid robot are revolute; all links are perfect rigid bodies, with known mass properties (total mass, center of mass, rotational inertia); each joint carries a motor; and there are no closed kinematic loops (i.e., there is only one way to go from any link of the robot to any other link). This latter fact means that the topology of the humanoid robot is a *tree* (Fig. 10.1): one of the rigid bodies in the trunk is the root, and the head, the hands and the feet are the leafs. Note, however, that *any* node in a tree structure can be chosen as root! The "bookkeeping" of node numbers changes when the root changes, and the recursive algorithms of the next Sections will traverse the robot structure differently.

Tree structures are interesting, because their dynamics algorithms are straightforward extensions of those for serial structures. It is well known that all nodes in a tree can be numbered in such a way that the root gets the lowest number, and the route from the root to any node n passes only through nodes with lower numbers $k < n$ (Fig. 10.1). When two nodes are considered on the same path from the root to a leaf, then the node with the lowest number is called the *proximal* node, and the other is the *distal* joint.

The algorithms in this text are presented for tree structures with only revolute joints. But they are easy to extend to other types of joints (prismatic, spherical, or Cardan joints), unactuated joints, or closed kinematic loops.

10.4 Frames—Coordinates—Transformation

Newton's law $\boldsymbol{f} = m\boldsymbol{a}$ is the foundation of in general dynamics, and hence also of robot dynamics. Newton's law is a relationship between force and acceleration *vector*. In order to be able to *calculate* with these physical vectors, one needs their *coordinates* with respect to known *reference frames*. One also needs to know how to *transform* the coordinate representations of the same physical vectors expressed in different reference frames. The following subsections introduce (i) reference frames that are adapted to the mechanical structure of the (humanoid) robot, (ii) six-dimensional rigid body forces and their coordinate transformations, and (iii) six-dimensional rigid body velocities and their coordinate transformations.

10.4.1 Frames

Figure 10.2 shows a typical link in a humanoid robot, together with its basic reference frames. The link i is connected to *one single* proximal link $i-1$ by a revolute joint with axis along $\boldsymbol{z}_{p_i}$; *several* distal links $i+1, \ldots, i+k$ can be attached to it, by joints along the vectors $\boldsymbol{z}_{d_{i+1}}, \ldots, \boldsymbol{z}_{d_{i+k}}$. Without loss of generality, we assume that the joint axes are the Z axes of orthogonal reference frames $\{p_i\}$ and $\{d_{i+1}\} \ldots \{d_{i+k}\}$, with origins on the joint axes. If the link is a leaf node in the robot, the distal "joint frames" are the user-defined *end-effector frames*. An "end-effector frame" is any frame on the humanoid robot that is of interest to the user; typical end-effectors are the feet, the hands, and the head (or rather, its ears and eyes).

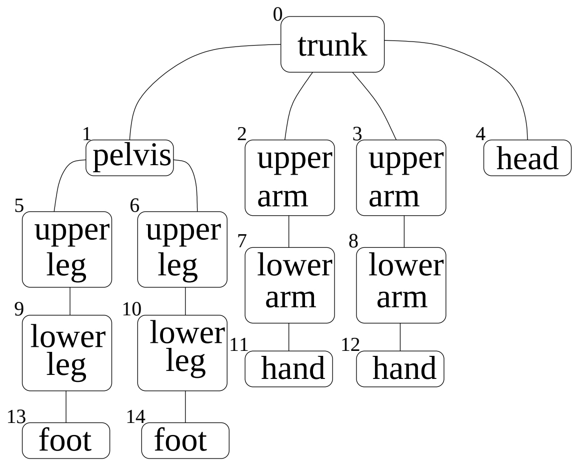


Figure 10.1: Tree topology of a typical humanoid robot. The nodes are rigid bodies, the edges are joints. (This schematic picture makes no claim whatsoever towards completeness!)

Since the links are rigid, the relative position and orientation of $\{d_{i+1}\} \dots \{d_{i+k}\}$ with respect to $\{p_i\}$ are constant. The link connected at $\{d_{i+1}\}$ has its "proximal" frame coinciding with $\{d_{i+1}\}$, up to a rotation about $\boldsymbol{z}_{d_{i+1}}$, over an angle $q_{d_{i+1}}$; this angle is *measured.* Its first and second time derivatives $\dot{q}_{d_{i+1}}$ and $\ddot{q}_{d_{i+1}}$ are also assumed to be measured, either directly, or by numerical differentiation of $q_{d_{i+1}}$ which is most often the case in practice.

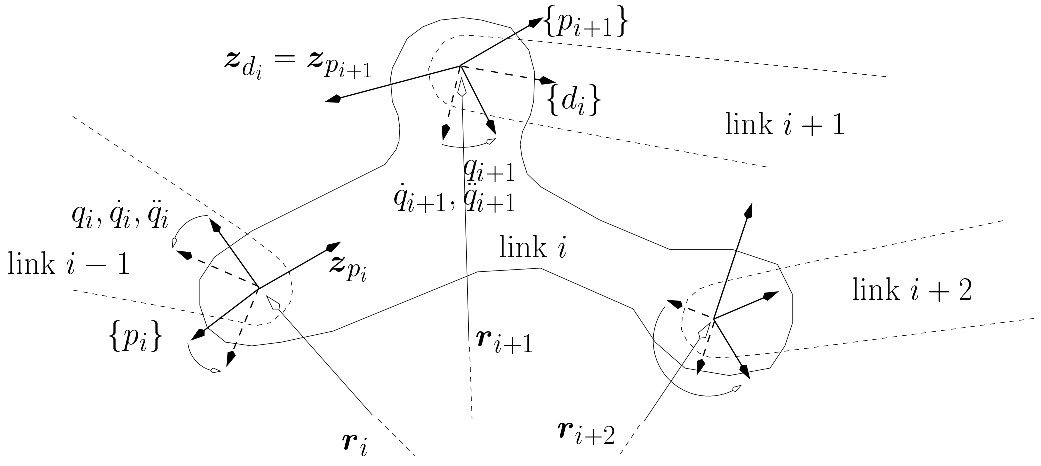


Figure 10.2: Reference frames and notation for links in tree-structured robot. "*p*" stands for "proximal," and "*d*" for distal.

10.4.2 Force/torque transformation

A *point mass* can feel linear forces only (represented by three-dimensional vectors $\boldsymbol{f}$), while a *rigid body* can feel both forces $\boldsymbol{f}$ and moments $\boldsymbol{m}$, represented by a six-dimensional coordinate vector $\boldsymbol{F} = (\boldsymbol{f}, \boldsymbol{m})$ (called a *wrench* in the Kinematics Chapter). Every set of such forces and torques is equivalent to one single force and one single torque applied at a given point of the rigid body. If such a resultant force $\boldsymbol{f}_2$ and torque $\boldsymbol{m}_2$ are known at a point "2," it is easy to find an equivalent set $(\boldsymbol{f}_1, \boldsymbol{m}_1)$ at another point "1":

$$\boldsymbol{f}_1 = \boldsymbol{f}_2, \qquad \boldsymbol{m}_1 = \boldsymbol{m}_2 + \boldsymbol{r}_{1,2} \times \boldsymbol{f}_2. \tag{10.1}$$

These equations considers physical vectors only. To represent the *coordinates* of the force and torque vectors acting in frame $\{i\}$, with respect to an absolute world frame, this text uses the notation $\boldsymbol{F}_i$:

$$\boldsymbol{F}_i = \begin{pmatrix} \boldsymbol{f}_i \\ \boldsymbol{m}_i \end{pmatrix}. \tag{10.2}$$

The notation for the coordinates is the same as for the physical vectors, because the interpretation will always be clear from the context. The transformation of force and torque coordinates from frame $\{2\}$ to frame $\{1\}$ involves the coordinates $(\boldsymbol{r}_{1,2}, {}^2_1\boldsymbol{R})$ of frame $\{2\}$ with respect to $\{1\}$:

$$\boldsymbol{F}_1 = \boldsymbol{\mathcal{T}}^F_{1,2}\, \boldsymbol{F}_2, \quad \text{with} \quad \boldsymbol{\mathcal{T}}^F_{1,2} = \begin{pmatrix} {}^2_1\boldsymbol{R} & 0_{3\times 3} \\ \widehat{\boldsymbol{r}}_{1,2}\, {}^2_1\boldsymbol{R} & {}^2_1\boldsymbol{R} \end{pmatrix}. \tag{10.3}$$

$\boldsymbol{\mathcal{T}}^F_{1,2}$ is the 6×6 force transformation matrix. $\widehat{\boldsymbol{r}}$ is the 3×3 matrix that represents taking the cross product with the vector $\boldsymbol{r}$ (expressed in frame $\{1\}$):

$$\widehat{\boldsymbol{r}} = \begin{pmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{pmatrix}. \tag{10.4}$$

Because of the orthogonality of $\boldsymbol{R}$, and the anti-symmetry of $\widehat{\boldsymbol{r}}$, the inverse transformation $\boldsymbol{\mathcal{T}}^F_{2,1} = (\boldsymbol{\mathcal{T}}^F_{1,2})^{-1}$ is simple:

$$\boldsymbol{\mathcal{T}}^F_{2,1} = \begin{pmatrix} {}^2_1\boldsymbol{R}^T & 0_{3\times 3} \\ -{}^2_1\boldsymbol{R}^T\, \widehat{\boldsymbol{r}}_{1,2} & {}^2_1\boldsymbol{R}^T \end{pmatrix}. \tag{10.5}$$

10.4.3 Velocity/acceleration transformation

A *point mass* can have a translational velocity only, represented by a three-dimensional vector $\boldsymbol{v}$, and a translational acceleration, $\boldsymbol{a} = \dot{\boldsymbol{v}}$. However, the velocity of a *rigid body* contains both translational and angular velocity components, $\boldsymbol{v}$ and $\boldsymbol{\omega}$; similarly for the body's acceleration: $\boldsymbol{a} = \dot{\boldsymbol{v}}$ and $\dot{\boldsymbol{\omega}}$. The transformation of velocities and accelerations between reference frames are similar (but not equal!) to those of forces. In physical vector form, this gives:

$$\boldsymbol{\omega}_1 = \boldsymbol{\omega}_2, \qquad \boldsymbol{v}_1 = \boldsymbol{v}_2 + \boldsymbol{r}_{1,2} \times \boldsymbol{\omega}_2. \tag{10.6}$$

This text uses $\dot{\boldsymbol{X}}_i$ to denote linear and angular velocity coordinates of frame $\{i\}$ with respect to an absolute world frame:

$$\dot{\boldsymbol{X}}_i = \begin{pmatrix} \boldsymbol{v}_i \\ \boldsymbol{\omega}_i \end{pmatrix}. \tag{10.7}$$

(Note that the *order* of the linear and angular three-dimensional vectors $\boldsymbol{v}$ and $\boldsymbol{\omega}$ is *arbitrary.* Making the alternative choice, $\dot{\boldsymbol{X}}_i = \begin{pmatrix} \boldsymbol{\omega}_i \\ \boldsymbol{v}_i \end{pmatrix}$ implies that the coordinate transformation formulae of the following paragraphs have to be changed accordingly.) The coordinate form of Eq. (10.6) is:

$$\dot{\boldsymbol{X}}_1 = \boldsymbol{\mathcal{T}}^V_{1,2}\, \dot{\boldsymbol{X}}_2, \quad \text{with} \quad \boldsymbol{\mathcal{T}}^V_{1,2} = \begin{pmatrix} {}^2_1\boldsymbol{R} & \widehat{\boldsymbol{r}}_{1,2}{}^2_1\boldsymbol{R} \\ 0_{3\times 3} & {}^2_1\boldsymbol{R} \end{pmatrix}. \tag{10.8}$$

$\boldsymbol{\mathcal{T}}^V_{1,2}$ is the 6×6 velocity transformation matrix; it contains the same 3×3 blocks as the force transformation matrix $\boldsymbol{\mathcal{T}}^F_{1,2}$ in Eq. (10.3). Note that we prefer notational convention over physical exactness: $\dot{\boldsymbol{X}}$ is strictly speaking *not* the time derivative of the position/orientation coordinates $\boldsymbol{X}$, but we use this notation because of its suggestive similarity with the point mass case. The same transformation as for velocities holds for accelerations too:

$$\ddot{\boldsymbol{X}}_1 = \boldsymbol{\mathcal{T}}^V_{1,2}\, \ddot{\boldsymbol{X}}_2, \quad \text{with} \quad \ddot{\boldsymbol{X}} = \begin{pmatrix} \dot{\boldsymbol{v}} \\ \dot{\boldsymbol{\omega}} \end{pmatrix}. \tag{10.9}$$

Again, the inverse transformation is simple:

$$\boldsymbol{\mathcal{T}}^V_{2,1} = \left(\boldsymbol{\mathcal{T}}^V_{1,2}\right)^{-1} = \begin{pmatrix} {}^2_1\boldsymbol{R}^T & -{}^2_1\boldsymbol{R}^T\, \widehat{\boldsymbol{r}}_{1,2} \\ 0_{3\times 3} & {}^2_1\boldsymbol{R}^T \end{pmatrix}. \tag{10.10}$$

Note that

$$\boldsymbol{\mathcal{T}}^F_{2,1} = \left(\boldsymbol{\mathcal{T}}^V_{1,2}\right)^T, \quad \boldsymbol{\mathcal{T}}^V_{2,1} = \left(\boldsymbol{\mathcal{T}}^F_{1,2}\right)^T. \tag{10.11}$$

10.4.4 Parametric uncertainty

The above-mentioned coordinate representations and transformations make implicit use of a lot of *geometrical parameters* of the mechanical robot structure, i.e., the relative positions and orientations of the robot's joint axes. These parameters are in general only known *approximately*, such that they should be considered as *uncertain parameters* in the robot model, for which (on-line or off-line) estimation techniques should be used.

10.5 Dynamics of a single rigid body

Newton's law $\boldsymbol{f} = m\boldsymbol{a}$ describes the dynamics of an *unconstrained point mass*. Any textbook on dynamics, e.g., [2], shows how to derive the motion law for a *rigid body*, i.e., a set of rigidly connected point masses. This derivation is straightforward (albeit algebraically a bit tedious): apply Newton's law to each "infinitesimal volume" of mass in the rigid body, and take the integral over the whole body. We just summarize the results here, and stress the important property that the dynamics are *linear* in the *external force* $\boldsymbol{F} = (\boldsymbol{f}, \boldsymbol{m})$, the *acceleration* $\ddot{\boldsymbol{X}} = (\boldsymbol{a}, \dot{\boldsymbol{\omega}})$, and the *mass matrix* (or, *inertia*) $\boldsymbol{M} = (m, \boldsymbol{I})$, but *nonlinear* in the *velocity* $\dot{\boldsymbol{X}} = (\boldsymbol{v}, \boldsymbol{\omega})$:

$$\boldsymbol{F} = \boldsymbol{M}\,\ddot{X} + \boldsymbol{F}^b \qquad \text{with} \quad \boldsymbol{F}^b = \begin{pmatrix} \boldsymbol{\omega} \times (\boldsymbol{\omega} \times m\boldsymbol{r}^c) \\ \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} \end{pmatrix}. \tag{10.12}$$

$\boldsymbol{F}^b$ is the so-called *bias force*, i.e., the force not due to acceleration, but to the current (angular) velocity. m is the total mass of the body. $\boldsymbol{r}^c$ is the vector from the origin of the reference frame in which all quantities are expressed to the centre of mass of the body. $\boldsymbol{I}$ is the 3×3 *angular inertia matrix* of the rigid body with respect to the current reference frame. Note that the bias force vanishes when (i) the centre of mass lies in the origin of the reference frame, *and* (ii) the body is spherical, i.e., its inertia $\boldsymbol{I}$ is a multiple of the unit matrix. Otherwise, the angular velocity generates a force due to the unbalance in the body. For example, assume you spin around a vertical axis through your body, while holding a heavy object in your hand. The object will not only be accelerated around the spin axis, but you will also feel a so-called "*centripetal*" force that tries to move the object away from you.

Usually, one knows the inertia $\boldsymbol{I}_c$ with respect to the centre of mass; the relationship between $\boldsymbol{I}_c$ and the inertia $\boldsymbol{I}$ at an arbitrary reference frame is:

$$\boldsymbol{I} = \boldsymbol{I}_c - m\widehat{\boldsymbol{r}}^c\widehat{\boldsymbol{r}}^c. \tag{10.13}$$

$\widehat{\boldsymbol{r}}^c$ is the 3×3 vector product matrix corresponding to $\boldsymbol{r}^c$; $\boldsymbol{r}^c$ is zero in a reference frame with origin in the centre of mass. Equation (10.13) shows that $\boldsymbol{I}$ is always a *symmetric* matrix. The 6×6 matrix $\boldsymbol{M}$ is the *generalized mass matrix*:

$$\boldsymbol{M} = \begin{pmatrix} m\mathbf{1}_{3\times 3} & m\widehat{\boldsymbol{r}}^c \\ -m\widehat{\boldsymbol{r}}^c & \boldsymbol{I} \end{pmatrix}. \tag{10.14}$$

The frame transformation properties of $\boldsymbol{I}$ and $\boldsymbol{M}$ are straightforwardly derived from the transformations of velocities, forces, and accelerations:

$$\boldsymbol{I}_1 = {}^2_1\boldsymbol{R}\,\boldsymbol{I}_2\,{}^2_1\boldsymbol{R}^T, \quad \boldsymbol{M}_1 = \boldsymbol{\mathcal{T}}^F_{1,2}\,\boldsymbol{M}_2\,(\boldsymbol{\mathcal{T}}^F_{1,2})^T. \tag{10.15}$$

For example, the latter follows from the relationships in Eq. (10.11) and from:

$$\begin{aligned} \boldsymbol{f}_2 = \boldsymbol{M}_2\,\boldsymbol{a}_2 \Rightarrow \boldsymbol{\mathcal{T}}^F_{1,2}\boldsymbol{f}_2 &= \boldsymbol{\mathcal{T}}^F_{1,2}\boldsymbol{M}_2\left((\boldsymbol{\mathcal{T}}^V_{1,2})^{-1}\boldsymbol{\mathcal{T}}^V_{1,2}\right)\boldsymbol{a}_2, \\ &\Rightarrow \boldsymbol{f}_1 = \boldsymbol{\mathcal{T}}^F_{1,2}\boldsymbol{M}_2(\boldsymbol{\mathcal{T}}^V_{1,2})^{-1}\boldsymbol{a}_1, \\ &\Rightarrow \boldsymbol{M}_1 = \boldsymbol{\mathcal{T}}^F_{1,2}\boldsymbol{M}_2(\boldsymbol{\mathcal{T}}^V_{1,2})^{-1}. \end{aligned}$$

Parametric uncertainty. The mass matrix and its coordinate representations and transformations also make implicit use of the same *geometrical parameters* of the mechanical robot structure as mentioned in Section 10.4.4, in addition to the three coordinates $\boldsymbol{r}^c$ of the centre of mass, the total mass m of the body, and the six parameters in the (symmetric) inertia matrix $\boldsymbol{I}$. Again, these parameters are in general only known *approximately*, such that they should be considered as *uncertain parameters* in the robot model, which require (on-line or off-line) estimation techniques.

10.6 Mass, acceleration, and force projections

Figure 10.3 shows the basic building block of every robot: one link of a robot connected to another link through a (revolute) joint. Forces act on both links, and these forces are related to the links' accelerations through their inertial properties. This Section explains

- how much of the acceleration of a proximal link is transmitted to its distal link;
- how much of the mass matrix of the distal link is felt by the proximal link;
- and how much of the force acting on the distal link is transmitted to the proximal link.

10.6.1 Inward mass matrix projection

The relationship between the acceleration $\ddot{\boldsymbol{X}}_1$ and the force $\boldsymbol{F}_1$ of link 1 for an unconstrained link 1 is given by the link's mass matrix $\boldsymbol{M}_1$. However, if link 1 is connected to link 2, the force $\boldsymbol{F}_1$ is not completely available to accelerate link 1; or, in other words, it seems as if it has become "heavier." This subsection explains how to find this so-called *articulated inertia* $\boldsymbol{M}^a_1$ of link 1, i.e., the mapping from the acceleration of the link to the corresponding force, taking into account the influence of the distal link. So, assume that link 1 is given an acceleration $\ddot{\boldsymbol{X}}_1$. In order to execute this acceleration, a force $\boldsymbol{F}_1$ is needed. This force is partially used to accelerate link 1 as if it were unconstrained, and a part $\boldsymbol{F}_2$ of the force $\boldsymbol{F}_1$ is transmitted through the revolute joint and causes an acceleration $\ddot{\boldsymbol{X}}_2$ of link 2. Both accelerations can only differ in their component about the common joint axis:

$$\ddot{\boldsymbol{X}}_1 - \ddot{\boldsymbol{X}}_2 = \boldsymbol{Z}\ddot{q}_2, \tag{10.16}$$

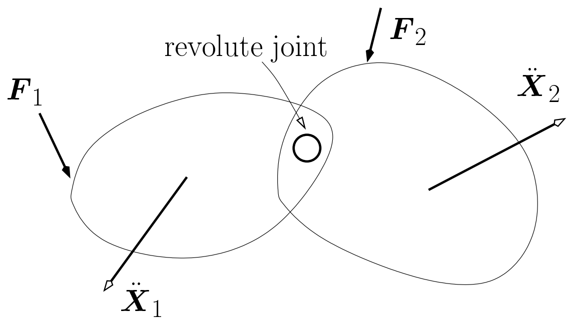


Figure 10.3: A rigid body is connected to another rigid body by a revolute joint. The joint cannot transmit a pure torque component about its axis, generated by the external forces.

with $\boldsymbol{Z}$ the six-dimensional basis vector of the joint, and $\ddot{q}$ the (as yet unknown) acceleration of the joint. (In a frame with its Z axis on the joint axis, $\boldsymbol{Z}$ has the simple coordinate representation $(0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1)^T$.) The transmitted force $\boldsymbol{F}_2$ cannot have a component about the revolute joint axis, hence:

$$\boldsymbol{Z}^T\boldsymbol{F}_2 = 0. \tag{10.17}$$

Because $\boldsymbol{F}_2 = \boldsymbol{M}_2\ddot{\boldsymbol{X}}_2$, with $\boldsymbol{M}_2$ the mass matrix of link 2, one finds that:

$$\boldsymbol{Z}^T\boldsymbol{M}_2\ddot{\boldsymbol{X}}_1 = \boldsymbol{Z}^T\boldsymbol{M}_2\boldsymbol{Z}\ddot{q}_2, \tag{10.18}$$

and

$$\ddot{q}_2 = \left(\boldsymbol{Z}^T\boldsymbol{M}_2\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^T\boldsymbol{M}_2\ddot{\boldsymbol{X}}_1. \tag{10.19}$$

Hence, the force $\boldsymbol{F}_1$ needed to accelerate link 1 by an amount $\ddot{\boldsymbol{X}}_1$ is given by

$$\boldsymbol{F}_1 = \boldsymbol{M}_1\ddot{\boldsymbol{X}}_1 + \left(\boldsymbol{M}_2 - \boldsymbol{M}_2\boldsymbol{Z}\left(\boldsymbol{Z}^T\boldsymbol{M}_2\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^T\boldsymbol{M}_2\right)\ddot{\boldsymbol{X}}_1, \tag{10.20}$$

$$= \boldsymbol{M}_1^a\ddot{\boldsymbol{X}}_1, \tag{10.21}$$

$$\text{with} \quad \boldsymbol{M}_1^a = \boldsymbol{M}_1 + \boldsymbol{M}_2 - \boldsymbol{M}_2\boldsymbol{Z}\left(\boldsymbol{Z}^T\boldsymbol{M}_2\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^T\boldsymbol{M}_2. \tag{10.22}$$

$\boldsymbol{M}_1^a$ the so-called *articulated body inertia*, [11], i.e., the increased inertia of link 1 due to the fact that it is connected to link 2 through an "articulation" which is the revolute joint. The mass of link 2 is "projected" onto link 1 through the joint between both links. The corresponding 6×6 *projection operator* $\boldsymbol{P}_2^{in}$ is:

$$\boldsymbol{P}_2^{in} = \boldsymbol{1} - \boldsymbol{M}_2\boldsymbol{Z}\left(\boldsymbol{Z}^T\boldsymbol{M}_2\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^T. \tag{10.23}$$

The superscript "*in*" stands for "inward," i.e., from distal link to proximal link. The matrix $\boldsymbol{P}_2^{in}$ is indeed a projection operator, because

$$\boldsymbol{P}_2^{in}\boldsymbol{P}_2^{in} = \boldsymbol{P}_2^{in}. \tag{10.24}$$

The total articulated inertia of link 1 is the sum of its own inertia $\boldsymbol{M}_1$ and the projected part $\boldsymbol{P}_2^{in}\boldsymbol{M}_2$ of the inertia of the second body:

$$\boldsymbol{M}_1^a = \boldsymbol{M}_1 + \boldsymbol{P}_2^{in}\boldsymbol{M}_2. \tag{10.25}$$

Let's take a closer look at Eq. (10.23). $\boldsymbol{Z}$ is a 6×1 vector; $\boldsymbol{M}_2$ is a 6×6 matrix. Hence, $\boldsymbol{Z}^T \boldsymbol{M}_2 \boldsymbol{Z}$ is a scalar, i.e., the element of $\boldsymbol{M}_2$ in the lower-right corner. And $\boldsymbol{M}_2 \boldsymbol{Z} \left(\boldsymbol{Z}^T \boldsymbol{M}_2 \boldsymbol{Z} \right)^{-1} \boldsymbol{Z}^T$ is the 6×6 matrix, which is a multiple of the last column of $\boldsymbol{M}_2$. Hence, the projection operator adds *more* than just link's 2 component of inertia about the joint axis, unless the mass of link 2 is symmetrically distributed about the joint axis.

10.6.2 Outward acceleration projection

The acceleration "transmitted" through the joint follows from Eq. (10.16):

$$\ddot{\boldsymbol{X}}_2 = \left(\mathbf{1} - \boldsymbol{Z} \left(\boldsymbol{Z}^T \boldsymbol{M}_2 \boldsymbol{Z} \right)^{-1} \boldsymbol{Z}^T \boldsymbol{M}_2 \right) \ddot{\boldsymbol{X}}_1, \tag{10.26}$$

$$= \left(\boldsymbol{P}_2^{in} \right)^T \ddot{\boldsymbol{X}}_1. \tag{10.27}$$

Hence, $\boldsymbol{P}_2^{out} = \left(\boldsymbol{P}_2^{in} \right)^T$ is the *outward* acceleration projector.

10.6.3 Inward force projection

Assume now that a force $\boldsymbol{F}_2$ acts on link 2. The question is how much of this force is transmitted through the joint between links 1 and 2. The naive answer to this question is to take the component $\boldsymbol{Z}^T \boldsymbol{F}_2$ along the joint axis, and subtract it from $\boldsymbol{F}_2$. The physical answer is as follows:

- $\boldsymbol{F}_2$ generates a torque $\boldsymbol{Z}^T \boldsymbol{F}_2$ about the joint axis.
- $\boldsymbol{Z}^T \boldsymbol{M}_2 \boldsymbol{Z}$ is the inertia of link 2 about the joint axis.
- $(\boldsymbol{Z}^T \boldsymbol{M}_2 \boldsymbol{Z})^{-1}$ is the corresponding acceleration generated by a unit torque about the joint axis.
- $\boldsymbol{Z}(\boldsymbol{Z}^T \boldsymbol{M}_2 \boldsymbol{Z})^{-1} \boldsymbol{Z}^T \boldsymbol{F}_2$ is the acceleration of link 2 caused by $\boldsymbol{F}_2$.
- This acceleration generates a six-dimensional force $\boldsymbol{M}_2 \boldsymbol{Z}(\boldsymbol{Z}^T \boldsymbol{M}_2 \boldsymbol{Z})^{-1} \boldsymbol{Z}^T \boldsymbol{F}_2$.

And this force is different from $\boldsymbol{Z}^T \boldsymbol{F}_2$: the mass of link 2 is in general not symmetrically distributed about the joint axis, such that an acceleration about the joint axis generates forces in all other directions too. The part $\boldsymbol{F}_1$ of the force $\boldsymbol{F}_2$ transmitted in inward direction to link 1 is then:

$$\boldsymbol{F}_1 = \boldsymbol{F}_2 - \boldsymbol{M}_2 \boldsymbol{Z}(\boldsymbol{Z}^T \boldsymbol{M}_2 \boldsymbol{Z})^{-1} \boldsymbol{Z}^T \boldsymbol{F}_2, \tag{10.28}$$

$$= \boldsymbol{P}_2^{in} \boldsymbol{F}_2. \tag{10.29}$$

Parametric uncertainty. The above-mentioned projections of forces, accelerations and inertias through a joint make use of the position and orientation parameters of the joint, as well as of the mass matrix of the links. As before, these parameters should be assumed to be uncertain, and hence are to be estimated and or adapted by the robot controller.

10.7 Link-to-link recursions

The dynamics algorithms for serial and humanoid robots can achieve *linear-time* complexity, because they use *inward and outward recursions* from link to link. These recursions propagate force, velocity, acceleration, and inertia from the root to the leafs (*outward* recursion), or vice versa (*inward* recursion). After each recursion step towards link i, all physcial properties of interest are expressed in the proximal frame $\{p_i\}$ of that link. The terminology "inward" and "outward" is unambiguous only for classical robot arms: their root is fixed in the environment, and they have a well-defined end-effector on which forces are applied, or motion constraints are acting. Humanoid robots, however, change their "fixed" root from one foot to the other when walking, and sometimes they have none of their feet on the ground. A humanoid robot can climb (especially in outer space where gravity is absent), such that one of its hands serves as the fixed root. And, as said already before, its topology allows any of its link to be root of its tree structure.

The physical properties of the recursions of dynamic parameters have already been discussed in the previous Sections; this Section basically adds only the somewhat involved "bookkeeping" of all coordinate representations involved in the recursions between different frames. Every recursion typically consists of three steps: for example, an outward recursion from link i to link $i+1$ first performs a coordinate transformation of the physical properties from the proximal frame $\{p_i\}$ of link i to its distal frame $\{d_{i+1}\}$; there the contribution of the joint q_{i+1} (position, velocity, ...) is taken into account; and the result is propagated to the proximal frame $\{p_{i+1}\}$ of the next link $i+1$. Of course, for efficiency reasons, linear-time algorithms try to keep these three steps as efficient as possible. That is the reason to choose coinciding proximal and distal frames on subsequent links.

10.7.1 Outward position recursion

This has already been discussed in the Kinematics chapter: the position and orientation of the end-effector frame are found from the measured joint angles by a multiplication of homogeneous transformation matrices that depend on the kinematic parameters of the robot.

10.7.2 Outward velocity recursion

The velocity recursion finds the linear and angular velocity $\dot{\boldsymbol{X}}_{i+1} = (\boldsymbol{v}_{i+1}, \boldsymbol{\omega}_{i+1})$ of the proximal frame $\{p_{i+1}\}$ on link $i+1$, given the linear and angular velocity $\dot{\boldsymbol{X}}_i = (\boldsymbol{v}_i, \boldsymbol{\omega}_i)$ of the proximal frame $\{p_i\}$ on link i, and given the joint angle speed $\dot{q}_{i+1}$ between both links. Because both links move only with respect to each other by a rotation about $\boldsymbol{z}_{d_{i+1}} = \boldsymbol{z}_{p_{i+1}}$, the following recursion equations are obvious:

$$\boldsymbol{v}_{i+1} = \boldsymbol{v}_i + \boldsymbol{\omega}_i \times \boldsymbol{r}_{i,i+1}, \tag{10.30}$$

$$\boldsymbol{\omega}_{i+1} = \boldsymbol{\omega}_i + \dot{q}_{i+1} \boldsymbol{z}_{d_{i+1}}. \tag{10.31}$$

with $\boldsymbol{r}_{i,i+1}$ the vector between the origins of the frames $\{i\}$ and $\{i+1\}$, i.e., $\boldsymbol{r}_{i,i+1} = \boldsymbol{r}_{i+1} - \boldsymbol{r}_i$. In coordinate form, the recursion $\dot{\boldsymbol{X}}_i \to \dot{\boldsymbol{X}}_{i+1}$ becomes:

$$\dot{\boldsymbol{X}}_{i+1} = \boldsymbol{\mathcal{T}}^V_{d_{i+1},p_{i+1}} \left(\boldsymbol{\mathcal{T}}^V_{p_i,d_{i+1}} \dot{\boldsymbol{X}}_i + \dot{q}_{i+1} \boldsymbol{Z}_{d_{i+1}} \right), \quad \text{with} \quad \boldsymbol{Z}_{d_{i+1}} = \begin{pmatrix} 0_{3\times 1} \\ \boldsymbol{z}_{d_{i+1}} \end{pmatrix}. \tag{10.32}$$

$\boldsymbol{\mathcal{T}}^V_{p_i,d_{i+1}}$ is the 6×6 velocity transformation matrix, Eq. (10.8), from the proximal to the distal frame on link i, and $\boldsymbol{\mathcal{T}}^V_{d_{i+1},p_{i+1}}$ transforms the velocities further to the proximal frame of the distal link. This last transformation is very simple, because the origins and the Z axes of both frames coincide.

10.7.3 Outward acceleration recursion

This recursion calculates the linear and angular acceleration $\ddot{\boldsymbol{X}}_{i+1} = (\dot{\boldsymbol{v}}_{i+1}, \dot{\boldsymbol{\omega}}_{i+1})$ of $\{p_{i+1}\}$, given the linear and angular acceleration $\ddot{\boldsymbol{X}}_i = (\dot{\boldsymbol{v}}_i, \dot{\boldsymbol{\omega}}_i)$ of $\{p_i\}$, and given the joint angle acceleration $\ddot{q}_{i+1}$. The recursion equations are found straightforwardly by taking the time derivative of the velocity recursion in Eq. (10.32):

$$\ddot{\boldsymbol{v}}_{i+1} = \dot{\boldsymbol{v}}_i + \dot{\boldsymbol{\omega}}_i \times \boldsymbol{r}_{i,i+1} + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{r}_{i,i+1}), \tag{10.33}$$

$$\dot{\boldsymbol{\omega}}_{i+1} = \dot{\boldsymbol{\omega}}_i + \ddot{q}_{i+1}\boldsymbol{z}_{d_{i+1}} + \boldsymbol{\omega}_i \times \dot{q}_{i+1}\boldsymbol{z}_{d_{i+1}}. \tag{10.34}$$

This uses the property that $\boldsymbol{\omega}_i \times \boldsymbol{x}$ is the time derivative of a vector $\boldsymbol{x}$ that is fixed to a body that rotates with an angular velocity $\boldsymbol{\omega}_i$. In coordinate form, the recursion $\ddot{\boldsymbol{X}}_i \rightarrow \ddot{\boldsymbol{X}}_{i+1}$ becomes:

$$\ddot{\boldsymbol{X}}_{i+1} = \boldsymbol{\mathcal{T}}^V_{d_{i+1},p_{i+1}} \left(\boldsymbol{\mathcal{T}}^V_{p_i,d_{i+1}} \ddot{\boldsymbol{X}}_i + \ddot{q}_{i+1}\boldsymbol{Z}_{d_{i+1}} + \boldsymbol{A}_{i+1} \right), \quad \text{with} \quad \boldsymbol{A}_{i+1} = \begin{pmatrix} \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \boldsymbol{r}_{i,i+1}) \\ \boldsymbol{\omega}_i \times \dot{q}_{i+1}\boldsymbol{z}_{d_{i+1}} \end{pmatrix}. \tag{10.35}$$

This acceleration recursion is identical to the velocity recursion of Eq. (10.32), except for the *bias acceleration* $\boldsymbol{A}_{i+1}$ due to the non-vanishing angular velocity $\boldsymbol{\omega}_i$.

10.7.4 Inward articulated mass recursion

The inward articulated mass matrix recursion calculates the articulated mass $\boldsymbol{M}^a_i$ of the proximal and distal links together, expressed in the proximal frame $\{p_i\}$ of the proximal link i, when the articulated mass $\boldsymbol{M}^a_{i+1}$ of the distal link $i+1$ is already known (expressed in its own proximal frame), as well as the mass matrix $\boldsymbol{M}_i$ of the proximal link (expressed in its proximal frame). Section 10.6.1 explained already how the mass matrix is transmitted through the revolute joint, Eq. (10.25). Hence, the coordinate form of the inward recursion $\boldsymbol{M}^a_i \leftarrow \boldsymbol{M}^a_{i+1}$ becomes:

$$\boldsymbol{M}^a_i = \boldsymbol{M}_i + \boldsymbol{\mathcal{T}}^F_{i,i+1} \left\{ \boldsymbol{P}^{in}_{i+1} \boldsymbol{M}^a_{i+1} \right\} \left(\boldsymbol{\mathcal{T}}^F_{i,i+1} \right)^T. \tag{10.36}$$

$\boldsymbol{M}^a_i$ is an operator working on the acceleration of link i, so, interpreted from right to left, one recognizes the following steps: (i) $(\boldsymbol{\mathcal{T}}^F_{i,i+1})^T = (\boldsymbol{\mathcal{T}}^V_{i,i+1})^{-1}$ transforms the coordinates of the acceleration to frame $\{p_{i+1}\}$; (ii) there it works on the part $\boldsymbol{P}^{in}_{i+1}\boldsymbol{M}^a_{i+1}$ of the articulated mass matrix of link $i+1$, and generates a force; and (iii) $\boldsymbol{\mathcal{T}}^F_{i,i+1}$ transforms the coordinates of this force back to link $\{p_{i+1}\}$. Note that (i) this recursion maintains the symmetry of the articulated mass matrix, and (ii) the force projection operator $\boldsymbol{P}^{in}_{i+1}$ uses the *articulated* mass matrix of link $i+1$, not its unconstrained mass matrix.

10.7.5 Inward force recursion

This section explains the recursion from $\boldsymbol{F}_{i+1}$, the total force felt by link $i+1$ at its proximal frame $\{p_{i+1}\}$, to $\boldsymbol{F}_i$, the total force felt by link i at its proximal frame $\{p_i\}$. $\boldsymbol{F}_i$ consists of two parts:

1. *Contributions from link $i+1$:*

   (a) The accumulated resultant total force $\boldsymbol{F}_{i+1}$.

   (b) The inertial force generated by the product of (i) the bias acceleration $\boldsymbol{A}_{i+1}$ of link $i+1$, Eq. (10.35) resulting from the angular velocity of link i, and (ii) the articulated mass matrix $\boldsymbol{M}^a_{i+1}$ of link $i+1$.

   (c) The joint torque τ_{i+1}.

   The sum of these forces is transmitted from link $i+1$ to link i, but only in part, due to the existence of the motion degree of freedom at the joint. The transmitted part of these force correspond to the transmitted force calculated in Eq. (10.29).

2. *Contributions from link i:*

(a) The velocity-dependent bias force $\boldsymbol{F}^b_i$, generated by the angular velocity and the mass properties of link i, Eq. (10.12).

(b) The "external force" $\boldsymbol{F}^e_i$, i.e., the resultant of all forces applied to link i, for example by people or objects pushing against it.

In coordinates, the recursion $\boldsymbol{F}_i \leftarrow \boldsymbol{F}_{i+1}$ becomes:

$$\boldsymbol{F}_i = \boldsymbol{\mathcal{T}}^F_{i,i+1}\,\boldsymbol{P}^{in}_{i+1}\,\boldsymbol{F}^{i+1}_{i+1} + \boldsymbol{F}^b_i + \boldsymbol{F}^e_i, \tag{10.37}$$

$$\text{with}\quad \boldsymbol{F}^{i+1}_{i+1} = \boldsymbol{F}_{i+1} + \boldsymbol{M}^a_{i+1}\boldsymbol{A}^b_{i+1} - \tau_{i+1}\boldsymbol{Z}_{i+1}. \tag{10.38}$$

The minus sign for the joint torque contribution comes from the fact that link i feels a torque $-\tau_{i+1}\boldsymbol{Z}_{i+1}$ if the motor at joint $i+1$ applies a torque of $+\tau_{i+1}$ units. This recursion can be slightly simplified: the joint torque vector $\tau_{i+1}\boldsymbol{Z}_{i+1}$ need not be constructed, because it gets operated on by the $\boldsymbol{Z}^T$ in $\boldsymbol{P}^{in}_{i+1}$, Eq. (10.23), which results in τ_{i+1} again.

10.8 Dynamics of serial arm

This Section applies the material of all previous Sections to construct linear-time algorithms for the forward and inverse dynamics of a serial robot arm. The algorithms are valid for arms with an arbitrary number of joints.

10.8.1 Inverse dynamics of serial arm

The acceleration of the end-effector link is specified by the user. For simplicity, assume that this end-effector acceleration has been transformed already into joint angle accelerations. (For robots with less or more than six joints, this transformation can be non-trivial and/or non-unique.) The joint torques needed to achieve this acceleration are then calculated as follows:

1. *Outward motion recursion.* Position, velocity, and bias acceleration due to angular velocities. The recursion is initialized with the position and velocity of the base.

2. *Inward articulated mass matrix recursion.* Initialized by $\boldsymbol{M}^a_N = 0$ for the end-effector link (which has number "N").

3. *Inward force recursion.* While performing this recursion, the joint torques τ_i in Eq. (10.37) are put to zero; the result of the recursion is the total load τ^l_i to be generated by the ith joint torque:

$$\tau^l_i = \boldsymbol{Z}^T_i(\boldsymbol{F}^b_i + \boldsymbol{M}^a_i\ddot{\boldsymbol{X}}^b_i). \tag{10.39}$$

   The acceleration $\ddot{\boldsymbol{X}}_{i-1}$ generated by the previous joint is not yet known at this stage of the ID algorithm.

4. *Outward joint torque recursion:*

$$\tau_i = \boldsymbol{Z}^T_i\left(\boldsymbol{F}^b_i + \boldsymbol{M}^a_i(\ddot{\boldsymbol{X}}^b_i + \ddot{\boldsymbol{X}}_{i-1})\right). \tag{10.40}$$

   This recursion uses the forward acceleration recursion, to calculate $\ddot{\boldsymbol{X}}_{i-1}$.

10.8.2 Forward dynamics

The acceleration generated by given joint torques can be found as soon as each joint knows which (articulated) mass it has to accelerate, and what external forces it has to withstand. The following scheme achieves this goal:

1. *Outward motion recursion.* Same as for ID.
2. *Inward articulated mass matrix recursion.* Same as for ID.
3. *Inward force recursion.* This recursion calculates the influence of inertial and external forces on each link, Eq. (10.37), starting with the end-effector link.
4. *Outward acceleration recursion.* The first joint now knows what (articulated) mass is attached to it, as well as all forces working on it (except for the joint torques), such that the acceleration generated by the joint torque τ_1 can be calculated; this acceleration is then used to find the bias acceleration for the second joint, and so on. The corresponding forward recursion of the *joint* acceleration is
$$\ddot{q}_i = (\boldsymbol{Z}_i^T \boldsymbol{M}_i^a \boldsymbol{Z}_i)^{-1} \left\{ \tau_i - \boldsymbol{Z}_i^T \left(\boldsymbol{F}_i^b + \boldsymbol{M}_i^a (\ddot{\boldsymbol{X}}_i^b + \ddot{\boldsymbol{X}}_{i-1}) \right) \right\}. \qquad (10.41)$$
Equation (10.35) gave already the corresponding outward recursion for the *spatial* accelerations $\ddot{\boldsymbol{X}}_i$. Gravity is taken into account by initializing the recursion with the gravitational acceleration: $\ddot{\boldsymbol{X}}_0 = \boldsymbol{g}$.
5. *Integration of joint accelerations.* The details of numerical integration algorithms are not discussed in this text.

10.9 Analytical form of serial chain dynamics

The previous Sections presented *recursive* algorithms. This means that the relationship between joint forces $\boldsymbol{\tau}$ and joint accelerations $\ddot{\boldsymbol{q}}$ (or end-effector acceleration $\ddot{\boldsymbol{X}}$) is not made explicit. Such an explicit *analytical* form of the dynamics would be very inefficient to calculate, since many terms are repeated. Nevertheless, an analytical form is interesting because it gives more insight: Are all relationships nonlinear, or do some relationships exhibit linear behaviour? What terms are important, and what others can be neglected in specific cases?

A closer inspection of the recursion relations reveals a general analytical form for the dynamics: the accelerations enter linearly in the dynamic equations; the velocities enter non-linearly due to the bias forces and accelerations; the influence of the gravity enters linearly. (These linearities will be very helpful to limit the complexity of estimation algorithms for the dynamic parameters of the robot.) Hence, the relationship between the joint forces $\boldsymbol{\tau}$, joint positions $\boldsymbol{q}$, joint velocities $\dot{\boldsymbol{q}}$, and joint accelerations $\ddot{\boldsymbol{q}}$ of a serial kinematic chain is of the following analytical form:
$$\boxed{\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{c}(\dot{\boldsymbol{q}}, \boldsymbol{q}) + \boldsymbol{g}(\boldsymbol{q}).} \qquad (10.42)$$
The matrix $\boldsymbol{M}(\boldsymbol{q})$ is called the *joint space mass matrix.* The vector $\boldsymbol{c}(\dot{\boldsymbol{q}}, \boldsymbol{q})$ is the vector of *Coriolis and centrifugal* joint forces. Some references write the vector $\boldsymbol{c}(\dot{\boldsymbol{q}}, \boldsymbol{q})$ as the product of a *matrix* $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and the vector $\dot{\boldsymbol{q}}$ of joint velocities. The vector $\boldsymbol{g}(\boldsymbol{q})$ is the vector of gravitational joint forces. In component form, Eq. (10.42) becomes
$$\tau_i = \sum_j \boldsymbol{M}_{ij}(\boldsymbol{q})\,\ddot{q}_j + \sum_{j,k} \boldsymbol{C}_{ijk}(\boldsymbol{q})\,\dot{q}_j \dot{q}_k + \boldsymbol{g}_i(\boldsymbol{q}). \qquad (10.43)$$
The joint gravity vector $\boldsymbol{g}(\boldsymbol{q})$ gives the joint forces needed to keep the robot in static equilibrium ($\dot{\boldsymbol{q}} = \ddot{\boldsymbol{q}} = 0$) under influence of gravity alone. The Coriolis and centrifugal vector gives the joint forces needed to keep the robot moving without acceleration or deceleration of the joints.

Joint space mass matrix. The joint space mass matrix $\boldsymbol{M}(\boldsymbol{q})$ gives the linear relationship between the joint forces and the resulting joint acceleration, *if* the robot is in rest, and *if* gravity is not taken into account. Hence, the physical meaning of $\boldsymbol{M}$ is that the ith column $\boldsymbol{M}_i(\boldsymbol{q})$ is the vector of joint forces needed to give a *unit* acceleration to joint i while keeping all other joints at *rest* (and after compensation for gravity).

It can be proven that the instantaneous *kinetic energy* T of the robot is given by

$$T = \dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}. \tag{10.44}$$

This has the same form as the expression $T = \frac{1}{2}mv^2$ for the kinetic energy of a point mass m moving with a velocity v.

10.10 Euler-Lagrange approach

The previous Sections started from Newton's law of motion to describe the dynamics of serial chains of rigid bodies. This approach is often called the *Newton-Euler* algorithm, and it uses the Cartesian velocities of all links in the chain, and the Cartesian forces exerted on all links. This involves a non-minimal number of variables, since each link has only *one* degree of freedom with respect to its neighbours, while the Cartesian velocities and forces for each link are six-vectors.

Another approach exists (the so-called *Euler-Lagrange* approach) that uses a *minimal number* of variables to describe the same dynamics. These *independent* variables are called *generalised coordinates*, [20]. In general, the minimal set of generalised coordinates might consist of coordinates that are not straightforwardly connected to the physical features of the system. However, for serial robots, the joint positions $\boldsymbol{q}$ are natural generalised coordinates for the *position* of the robot. The joint forces $\boldsymbol{\tau}$ are the corresponding *generalised forces.* The generalised velocities and accelerations of the system are simply the time derivatives of the joint coordinates, so no new independent variables are needed to describe the system's *dynamics.* Instead of Newton's laws, the Euler-Lagrange approach uses Hamilton's Principle (1834) as a starting point: A dynamical system evolves in time along the trajectory, from instant t_1 to instant t_2, that makes the *action integral*

$$I = \int_{t_1}^{t_2} \mathcal{L}\, dt \tag{10.45}$$

reach an *extremal* value (i.e., a local minimum or maximum), [14, 15, 22, 25, 28, 31, 32].

A problem of this kind is called a *variational problem.* The integrand $\mathcal{L}$ is called the *Lagrangian* of the dynamical system. Hamilton's Principle is *very general*, and applies to many more cases than just a robotic system of masses moving under the influence of forces, as considered in this text. For this latter case, the Lagrangian $\mathcal{L}$ is equal to the difference of the kinetic energy T of the system, and the potential energy V:

$$\mathcal{L} = T - V. \tag{10.46}$$

If forces that cannot be derived from a potential function (so-called *non-conservative* forces, such as joint torques, or friction) act on the system, then the Lagrangian is extended with one more energy term W, i.e., the work done by these forces:

$$\mathcal{L} = T - V + W. \tag{10.47}$$

William Rowan Hamilton's (1805–1865) Principle was the end-point of a long search for "minimal principles," that started with Fermat's *Principle of Least Time* (Pierre de Fermat (1601–1665), [7]) in optics, and Maupertuis' *Principle of Least Action* (Pierre Louis Moreau de Maupertuis (1698–1759), [8, 9]). The precise contents of the word "action" changed over time (Lagrange, for example, used the product of distance and momentum as the "action", [20]) until Hamilton revived the concept, and gave it the meaning it still has today, i.e., the product of energy and time.

This paragraph explains how one should interpret Hamilton's principle in the context of robot motion. Assume some forces act on the robot: gravity, joint forces, external forces on the end-effector or directly on intermediate links of the robot. The robot will perform a certain motion from time instant t_1 to time instant $t_2 > t_1$. This trajectory is fully deterministic *if* all parameters are known: the robot's kinematics, the mass matrices of all links, the applied forces, the instantaneous motion. This trajectory is "extremal" in the following sense: consider any alternative trajectory with the same start and end-point, for which (i) the same points are reached at the same start and end instants t_1 and t_2, (ii) the trajectory in between can deviate from, but remains "in the neighbourhood" of, the physical trajectory, and (iii) the same forces act on the robot. Then, the action integral (10.45) for the physically executed path is smaller than the action integral for *any* of the alternative paths in the neighbourhood.

Hamilton's principle is an *axiom*, i.e., it is stated as a fundamental physical principle, at the same footing as, for example, Newton's laws, or, in a different area of physics, the laws of thermodynamics. Hence, it was never derived from more fundamental principles. What *can* be proven is that different principles turn out to be equivalent, i.e., they lead to the same results. This is, for example, what Silver [29], did for the Newton-Euler approach of the previous Sections, and the Euler-Lagrange approach of this Section. The validity of Newton's laws and Hamilton's Principle as basic physical principles is corroborated by the fact that they gave the correct answers in all cases they could be applied to. From this "evidence" on a sample of characteristic problems, one has then *induced* their general validity to a whole field of physics. This means that these principles remain "valid" until refuted. Probably the two most famous refutations in the history of science are Copernicus' heliocentric model (refuting the geocentric model), and Einstein's Principles of Relativity (that replace Newton's laws at speeds close to the speed of light, or for physics at a cosmological scale.)

The interpretation of Hamilton's principle above assumed that one knows the physically executed path. However, in practice, this is exactly what one is looking for. So, how can Hamilton's principle help us to find that path? Well, the Swiss mathematician Leonhard Euler proved that the solution to the kind of variational problem that Hamilton used in his Principle, leads to a set of partial differential equations on the Lagrangian function, [10]. This transformation by Euler is valid independently of the fact whether or not one really knows the solution. Although Euler applied his method only to the particular case of a single particle, his solution approach is much more general, and is valid for the context of serial robot dynamics. Euler's results are also much more practical to work with than Hamilton's principle, since it reduces finding the physical trajectory to solving a set of partial differential equations (PDEs) with boundary conditions that correspond to the state of the system at times t_1 and t_2. So, in practice one starts from Euler's PDEs as "most fundamental" principle, instead of starting from Hamilton's principle.

The name of the French mathematician and astronomer Joseph Louis Lagrange (1736–1813) is connected to the method described in this Section because he was the first to apply the "principle of least action" to general dynamical systems. The equations of motion he derived for a system of rigid bodies are exactly Euler's PDEs, applied to mechanics. Euler's and Lagrange's contributions in the area of dynamics of point masses or rigid bodies date from more than half a century before Hamilton stated his Principle. However, Hamilton's Principle is more general than the dynamics that Euler and Lagrange considered in their work.

This Section on the Euler-Lagrange approach is much shorter than the Section on the Newton-Euler approach. This does not mean that the Euler-Lagrange approach is simpler or more practical; its shorter length is a mere consequence of the fact that most of the necessary material has already been introduced in the Newton-Euler Section, such as, for example, the expressions for the kinetic and potential energy of serial robots. The following paragraph will just describe how the well-known dynamical equations of Lagrange are derived from (i) Hamilton's Principle, and (ii) the Euler differential equations that solve the variational problem associated with the action integral.

10.10.1 Euler-Lagrange equations

This Section derives the Euler-Lagrange equations that describe the dynamics of a serial robot. We start from Hamilton's Principle, and apply Euler's solution approach to it. This derivation can be found in most classical textbooks on physics, e.g., [6, 12, 13, 28]. Assume that the extremum value of the action integral is given by

$$I = \int_{t_1}^{t_2} \mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)\, dt, \tag{10.48}$$

with $\mathcal{L} = T - V$ the desired Lagrangian function we are looking for. $\mathcal{L}$ is a function of the n generalised coordinates $\boldsymbol{q} = (q_1, \dots, q_n)$, and their time derivatives. Both $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ depend on the time. A *variation* of this integral is a function of the following form:

$$\Phi(\boldsymbol{\epsilon}) = \int_{t_1}^{t_2} \mathcal{L}(\boldsymbol{q} + \boldsymbol{\epsilon}^T \boldsymbol{r}, \dot{\boldsymbol{q}} + \boldsymbol{\epsilon}^T \dot{\boldsymbol{r}}, t)\, dt, \tag{10.49}$$

with $\boldsymbol{\epsilon}$ a vector of real numbers, and $\boldsymbol{r}$ a set of real functions of time that vanish at t_1 and t_2. Note that Φ is considered as a function of the epsilons, *not* of the generalised coordinates. Hence, this variation $\Phi(\boldsymbol{\epsilon})$ is a function that can approximate arbitrarily close the extremum $\mathcal{L}$ we are looking for if the epsilons are made small enough. Now, this function $\Phi(\boldsymbol{\epsilon})$ should reach an extremal value (corresponding to the extremal value of the action integral) for all $\epsilon_i = 0$. Hence, Φ's partial derivatives with respect to the ϵ_i should vanish at the values $\epsilon_1 = \cdots = \epsilon_n = 0$. Hence, also the following identity will be fulfilled:

$$0 = \epsilon_1 \left(\frac{\partial \Phi}{\partial \epsilon_1}\right)_{\epsilon_1=0} + \epsilon_2 \left(\frac{\partial \Phi}{\partial \epsilon_2}\right)_{\epsilon_2=0} + \cdots + \epsilon_n \left(\frac{\partial \Phi}{\partial \epsilon_n}\right)_{\epsilon_n=0} \tag{10.50}$$

$$= \int_{t_1}^{t_2} \left(\epsilon_1 \left(\frac{\partial \mathcal{L}}{\partial q_1} r_1 + \frac{\partial \mathcal{L}}{\partial \dot{q}_1} \dot{r}_1\right) + \epsilon_2 \left(\frac{\partial \mathcal{L}}{\partial q_2} r_2 + \frac{\partial \mathcal{L}}{\partial \dot{q}_2} \dot{r}_2\right) + \cdots + \epsilon_n \left(\frac{\partial \mathcal{L}}{\partial q_n} r_n + \frac{\partial \mathcal{L}}{\partial \dot{q}_n} \dot{r}_n\right)\right) dt. \tag{10.51}$$

The right-hand side is called the *first variation* of Φ, because it is formally similar to the first order approximation of a "normal" function, i.e., the first term in the function's Taylor series. Partial integration on the factors multiplying each of the ϵ_i gives

$$\epsilon_i \frac{\partial \mathcal{L}}{\partial \dot{q}_i} r_i \Bigg|_{t_1}^{t_2} + \epsilon_i \int_{t_1}^{t_2} r_i \left(\frac{\partial \mathcal{L}}{\partial q_i} - \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) dt. \tag{10.52}$$

The evaluation at the boundaries t_1 and t_2 vanishes, by definition of the functions r_i. Moreover, these functions are arbitrary, and hence the extremal value of the variation is reached when each of the factors multiplying these functions r_i becomes zero. This gives the *Euler-Lagrangian equations* for an unforced system (i.e., without external forces acting on it):

$$\boxed{\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = 0, \quad i = 1, \dots, n,} \qquad \text{or, in vector form,} \qquad \boxed{\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{q}} = \boldsymbol{0}.} \tag{10.53}$$

10.11 Newton-Euler vs. Euler-Lagrange

Since both the Newton-Euler approach and the Euler-Lagrange approach discuss the *same physical problem*, they must be equivalent, [29]. So, why would one prefer one method to the other? This question doesn't have a unique answer, since this answer depends on the context and the envisaged application. However, some general remarks can be made:

- *Recursive* Euler-Lagrange algorithms have been developed, such that the historical objection against using the Euler-Lagrange approach because of efficiency reasons has lost much (although not everthing) of its initial motivation.
- Hamilton's Principle is clearly independent of the mathematical representation used, hence the Euler-Lagrange equations derived from it are (by construction) *invariant* under any change of mathematical representation.
- The Newton-Euler method starts from the dynamics of all *individual* parts of the system; the Euler-Lagrange method starts from the kinetic and potential energy of the *total* system. Hence, the Euler-Lagrange approach is easier to extend to systems with infinite degrees of freedom, such as in fluid mechanics, or for robots with flexible links.
- The Newton-Euler method looks at the *instantaneous* or *infinitesimal* aspects of the motion; the Euler-Lagrange method considers the states of the system during a *finite* time interval. In other words, the Newton-Euler approach is *differential* in nature, the Euler-Lagrange approach is an *integral* method, [32].
- The Newton-Euler method uses *vector quantities* (Cartesian velocities and forces), while the Euler-Lagrange method works with *scalar quantities* (energies).

References for this Chapter

[1] V. I. Arnold, K. Vogtmann, and A. Weinstein. *Mathematical Methods of Classical Mechanics*, volume 60 of *Graduate Texts in Mathematics*. Springer Verlag, New York, NY, 2nd edition, 1989.

[2] H. Baruh. *Analytical Dynamics*. WCB McGraw-Hill, 1999.

[3] J. M. Cameron and W. J. Book. Modeling mechanisms with nonholonomic joints using the Boltzmann-Hamel equations. *Int. J. Robotics Research*, 16(1):47–59, 1997.

[4] A. Codourey. Dynamic modelling and mass matrix evaluation of the DELTA parallel robot for axes decoupling control. In *Int. Conf. Intel. Robots and Systems*, pages 1211–1218, Osaka, Japan, 1996.

[5] H. C. Corben and P. Stehle. *Classical Mechanics*. Dover Publications, Inc., Mineola, NY, 2nd edition, 1994.

[6] R. Courant and D. Hilbert. *Methoden der Mathematischen Physik I*. Springer Verlag, 1968.

[7] P. de Fermat. ?? In *Varia opera mathematica*. Culture et civilisation, Brussels, Belgium, 1679.

[8] P. L. M. de Maupertuis. Essai de cosmologie. In *Oeuvres, Tome I*. Olms, Hildesheim, Germany, 1759.

[9] P. L. M. de Maupertuis. Accord de différentes lois de la nature. In *Oeuvres, Tome IV*. Olms, Hildesheim, Germany, 1768.

[10] L. Euler. Methodus inveniendi lineas curvas maximi minimive proprietate gaudentes, sive solutio problematis isoperimetrici latissimo sensu accepti, additamentum II (1744). In C. Carathéodory, editor, *Opera omnia*, pages LII–LV, 298–308. Fussli, Zürich, Switzerland, 1952.

[11] R. Featherstone. *Robot dynamics algorithms*. Kluwer, Boston, MA, 1987.

[12] G. R. Fowles. *Analytical Mechanics*. Holt, Rinehart and Winston, New York, NY, 3rd edition, 1977.

[13] H. Goldstein. *Classical mechanics*. Addison-Wesley Series in Physics. Addison-Wesley, Reading, MA, 2nd edition, 1980.

[14] W. R. Hamilton. On a general method in dynamics. In *Philosophical Transactions of the Royal Society* [16], pages 247–308. Reprinted in Hamilton: The Mathematical Papers, Cambridge University Press, 1940.

[15] W. R. Hamilton. Second essay on a general method in dynamics. In *Philosophical Transactions of the Royal Society* [16], pages 95–144. Reprinted in Hamilton: The Mathematical Papers, Cambridge University Press, 1940.

[16] W. R. Hamilton, A. W. Conway, (ed.), and A. J. McConnell, (ed.). *The Mathematical Papers: Vol. II Dynamics.* Cunningham Memoir No. XIV. Cambridge University Press, London, England, 1940.

[17] W. W. Hooker. A set of r dynamical attitude equations for an arbitrary n-body satellite having r rotational degrees of freedom. *AIAA Journal*, 8(7):1205–1207, 1970.

[18] W. W. Hooker and G. Margulies. The dynamical attitude equations for an arbitrary n-body satellite having r rotational degrees of freedom. *J. Astronautical Sciences*, 12(4):123–128, 1965.

[19] Z. Ji. Dynamics decomposition for Stewart platforms. *Trans. ASME J. Mech. Design*, 116:67–69, 1994.

[20] J. L. Lagrange. Mécanique analytique. In J.-A. Serret, editor, *Oeuvres.* Gauthier-Villars, Paris, France, 1867.

[21] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul. Online computational scheme for mechanical manipulators. *Trans. ASME J. Dyn. Systems Meas. Control*, 102:69–76, 1980.

[22] J. E. Marsden and T. S. Ratiu. *Introduction to Mechanics and Symmetry*, volume 17 of *Texts in Applied Mathematics.* Springer Verlag, New York, NY, 1994.

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

[24] C. Reboulet and T. Berthomieu. Dynamic models of a six degree of freedom parallel manipulator. In *Int. Conf. Advanced Robotics*, Pisa, Italy, 1991.

[25] H. Rund. *The Hamilton-Jacobi theory in the calculus of variations: Its role in mathematics and physics.* The New University Mathematics Series. Van Nostrand, London, England, 1966.

[26] S. K. Saha and J. Angeles. Kinematics and dynamics of a three-wheeled 2-dof AGV. In *IEEE Int. Conf. Robotics and Automation*, pages 1572–1577, Scottsdale, AZ, 1989.

[27] S. K. Saha and J. Angeles. Dynamics of nonholonomic mechanical systems using a natural orthogonal complement. *Trans. ASME J. Appl. Mech.*, 58:238–243, 1991.

[28] F. A. Scheck. *Mechanics, from Newton's laws to deterministic chaos.* Springer Verlag, Berlin, Germany, 2nd edition, 1994.

[29] D. B. Silver. On the equivalence of Lagrangian and Newton-Euler dynamics for manipulators. *Int. J. Robotics Research*, 1(2):60–70, 1982.

[30] A. F. Vereshchagin. Computer simulation of the dynamics of complicated mechanisms of robot-manipulators. *Engineering Cybernetics*, 12(6):65–70, 1974.

[31] D. A. Wells. *Lagrangian dynamics.* Schaum's Outline Series. McGraw-Hill, New York, NY, 1967.

[32] W. Yourgrau and S. Mandelstam. *Variational principles in dynamics and quantum theory.* Pitman and Sons, London, England, 3rd edition, 1968.

[33] K. E. Zanganeh, R. Sinatra, and J. Angeles. Dynamics of a six-degree-of-freedom parallel manipulator with revolute legs. In *World Automation Congress WAC'96*, volume 3, Robotic and Manufacturing Systems, pages 817–822, Montpellier, France, 1996.

[34] H. Ziegler. *Mechanics*, volume 2. Addison-Wesley, Reading, MA, 1966.

Chapter 11

Motion planning

11.1 Introduction

Robots move things around (including themselves as in the case of mobile robots), create changes in the environment using their tools (grippers, welding guns or torches, paint pistols, etc.) or inspect the environment by means of sensors attached to their end-effectors. All these tasks have one thing in common: the robot is commanded to move to a sequence of desired positions and orientations in space. Often the user is also explicitly interested in *how* the robot gets there: he might want the robot to move through some specified *via-points* (taught manually, or generated by off-line or on-line task planners) with a prescribed speed, to follow a surface in the environment (making contact with this surface, or following it at a prescribed distance), or to optimise some performance criterions (minimum time, minimum distance, minimum energy consumption, avoidance of actuator saturation, minimum excitation of the robot's natural frequencies, avoidance of obstacles, etc.).

Fact-to-Remember 63 (Basic ideas of this Chapter)
This Chapter introduces some fundamental ideas in planning. Some methods are discrete, others are analytical. "Discrete" often means "faster," "less accurate," and "sub-optimal." Most discrete approaches use the concept of configuration space (also called C-space).

No really standardised terminology exists about what exactly is understood under the terms *motion*, *path* or *task*:

Task Examples are: assemble a video recorder; drive from street "A" in town "Aa" to street "B" in town "Ba"; paint a workpiece; etc. In general, a task consists of a (large) number of more primitive subtasks, each of which requires the robot to perform an action that can be specified (i.e., fully described) by one single "command." Each of these commands describes a certain *path* that the robot has to follow, as well as with what speeds it has to *move* along that path.

Path A (continuous) set of positions and orientations that the robot has to move through. Roughly speaking, the path contains *geometric* information only.

Typically, initial, intermediate and final poses and/or motion directions are given, and the *path planner* must link both poses in a continuous and "optimal" way.

Motion Instantaneous information about with what *velocity* the robot has to move along a given geometrical path, or based on instantaneous sensor information. The "large" distance between the start and end poses is subdivided in a lot of "infinitesimally separated" poses, which are generated at a pace synchronised with the robot's *motion controller*. The conversion of a geometric path into instantaneous motion commands is often called *motion interpolation*, or *trajectory generation*.

The difference between path and motion planning is not always really clear. Fargoing similarities exist with computer graphics and animation: in these fields, the artist creates "key frames" for all moving objects in the scene (i.e., a set of intermediate positions and orientations) and lets the computer interpolate between these key frames to generate "naturally looking" motions.

This Chapter does *not* discuss the planning of *why* the subgoal poses are needed, and *how* they are to be sequenced in order to execute a complete task. This task planning problem is difficult, to say the least, and no general body of solution techniques exists yet since task planning is so much task-dependent and (especially) requires levels of creativity that cannot be achieved by even the most advanced current-day software. Planning is indeed not an exact science, in the sense that no general "planning laws" exist to guide or constrain the planning activity. Hence, numerous "common sense" approaches have been developed. The first Section in this Chapter discusses the major trade-offs underlying this common sense. Different planning papers emphasize different aspects in these trade-offs; the results is a enormous amount of different planning approaches. The other Sections only present some of the most popular techniques in (a little bit) more detail.

11.2 Major trade-offs in planning

One can almost never find "the" best motion specification, either because it just doesn't exist (recall for example the arbitrariness existing in the definition of the distance between two poses), or because it costs too much to find it (for example, what would be the most optimal way to assemble an airplane?). Hence, all planning algorithms emphasize one or more sub-aspects of the problem. The following Sections highlight some of the most common trade-offs related to this focussing on sub-aspects.

11.2.1 Polynomial vs. non-polynomial trajectories

Polynomial functions interpolating between goal positions seem, at first sight, a natural approach to describe spatial robot trajectories. But one has to keep in mind that polynomials interpolating many via-points become, in general, very oscillatory, [51]. Hence, one more often uses "spline-like" approaches, in which locally defined path segments are glued together while obeying continuity constraints on position, velocity, acceleration, and/or jerk. (Jerk is the time derivative of acceleration.) The simplest approach is to use straight lines connected by smooth transition curves, [41, 50]; solutions for these transition curves are illustrated in many of the following Sections. During the transition, the path might deviate from the nominally specified trajectory when priority is given to making the transition smooth, not to making it pass exactly through the via-points or be tangent to the path at the via-points up to a certain order in a prescribed way.

11.2.2 Joint space vs. Cartesian space

The simplest approach to trajectory generation uses interpolation in *joint space*: the joint angle vectors corresponding to the start and end poses of the robot are calculated, and joint space trajectories between both are generated. This problem is much simpler than a full Cartesian space interpolation, for the following reasons:

1. Eventually, every motion specification is translated in a stream of desired joint motions. Chapter 7 already made clear that the transformation from Cartesian motion parameters into joint space motion parameters is not always uniquely defined. Moreover, most real robot joints have mechanical limits, that should be taken into account by the motion planner. Both of these problems are easier to solve in joint space.

2. Near singularities of the robot, the transformation from Cartesian to joint space motion parameters breaks down. Recall that no singularities (except mechanical limits) exist for paths defined in joint space.

3. A general Cartesian motion planner is confronted with the fact that no bi-invariant metric exists for the motion parameters of rigid bodies (Chap. 3). This means that the results possibly depend on the reference frames chosen on the robot's base and end-effector.

4. It might well be that parts of a desired Cartesian path do not belong to the workspace of the robot, but that this cannot be predicted from knowing only the initial and final Cartesian poses. This problem can be coped with easily in joint space.

Fact-to-Remember 64 (Joint space interpolation)
Roughly speaking, joint space interpolation reduces a six-dimensional, coupled motion planning problem to n $\underline{\text{one-dimensional}}$, $\underline{\text{uncoupled}}$ problems, n being the number of joints in the robot.

(The user could decide to (slightly) couple these n one-dimensional problems if he requires the motions for all n joints to take exactly the same amount of time.) The major drawback of joint space interpolation is the *unpredictability* of the resulting motion: since the transformation from joint space to Cartesian space is highly nonlinear. For example, it is difficult to guarantee collision-free Cartesian paths by using a joint space interpolator.

11.2.3 Sensor-based vs. sensor-less

Most of the planning literature deals with sensor-less approaches: a plan is generated off-line, and no provision is made to take sensor inputs into account during execution. This is the major reason why this literature is of not much use for "intelligent robots," that, by the nature of their tasks, need their sensors constantly. The sensor-less planning literature is useful for industrial applications, where the robots, its tools and its environment are well under control.

11.2.4 Model-based vs. model-free

Planning can, almost by definition, only take place in a context where a model of the robot and its environment is available. However, a rich literature exists on model-free planning, which is also called *reactive planning* since it typically boils down to giving the robot controller a set of rules that dictate how it should react to its sensor readings. Reactive plans are popular in collision avoidance. This text emphasized model-based planning only.

11.2.5 Local vs. global

Some planning techniques discuss local planning only: how should the robot move at the next instant in time, in order to move "in the direction" of the final goal. The redundancy resolution techniques based on the extended Jacobian (Sect. 7.14.2) are examples of locally optimal algorithms. Most mobile robot path planners have some global planning module available: it determines the route to follow, given a map of the world.

11.2.6 Robustness vs. time optimality

In many applications the robot programmer is most concerned about the spatial poses that the robot will move through, and less with at what time instants exactly the robot traverses the different intermediate poses generated by the trajectory planning algorithm. For example, if one wants a mobile robot to find its way through the corridors of a large building, one puts generally more emphasis on the fact whether or not the robot can reach its goal without collisions, than on the fact whether or not it could have done it a minute or two faster.

11.2.7 Velocity vs. acceleration vs. jerk constraints

All real-world robots are moved by actuators that can produce only limited torques. Hence, the trajectory planner for the robot should not generate motions that require torques beyond these limits. If no dynamic model for the robot is available, however, one cannot calculate the joint torques required for the motion; in this case, it is common practice to replace the joint torque limits by (conservative) limits on the velocity, the acceleration, and/or the jerk. Constraints on first-order or higher-order derivatives of the position require more complicated algorithms on the one hand, but can produce *smoother* motions on the other hand. The "optimum" in this trade-off cannot be defined straightforwardly .

The smoothness of the path is very important for mobile robots in order to avoid slipping of the wheels. For serial and parallel robots, smoothness is important because fastly changing or discontinuous acceleration and/or jerk increase the chance to excite the robot's or the tool's *natural frequencies.* This excitation has always a negative influence on the end-effector position of the robot; moreover, if these oscillations cannot be controlled, damage could be caused to the robot, its tools, or the environment. Serial and parallel manipulators cannot slip, so they are more robust against unsmooth trajectories. "Robust" in this context means that they can at least *measure* the deviations generated by unsmooth trajectories, except for the deviation components due to flexibility of joints and links.

Note that "smoothness" is not equivalent to "minimal excitation of natural freqencies": indeed, a perfect sine function with high frequency is infinitely smooth, but is very likely to excite the natural frequencies. Hence, smoothness is only part of the picture: the *variation* of the motion profile must also be maximally "bounded", i.e., its magnitude must remain small with respect to a given norm. As remarked already many times before in this text, no such natural norm exists for rigid body motions. On the other hand, it is unpractical to take all higher-order derivatives of the motion into account. Hence, a reasonable trade-off is to limit oneself to the *acceleration* or *jerk* levels.

11.2.8 Performance optimization vs. programming optimization

Even if the user possesses all data necessary to, for example, optimise the energy comsumption of the motion by using the complete dynamic model of the robot, he might opt for a programming approach that generates a trajectory that is less optimal during execution but that takes much less time to specify. Some commonly applied programming simplifications are: (i) decomposition of the Cartesian path into decoupled *translation* and *orientation* interpolation problems, (ii) specification of conservative limits on the allowable velocities and accelerations, instead of calculating them exactly from the dynamics of the robot, and (iii) manual teaching of via-points instead of optimization of these via-points on an off-line robot simulation system.

11.2.9 Object motion planning vs. robot motion planning

Much research in robotics focusses on the creation of plans to move a rigid body amongst obstacles in its environment. (This part of motion planning is particularly popular with computer science people working in robotics, since they often don't have access to, or experience with, real robotic devices...) This rigid body can be the robot itself (e.g., a mobile robot) or a workpiece connected to a serial or parallel manipulator. This kind of planning problem is often called the "*piano movers problem*," since human furniture movers have been confronted for centuries already with the difficulties of moving a grand piano out of a house. In robotics, one tries since about two decades to solve this piano problem, and several related motion planning problems, in an automated way. In this area of motion planning, artificial intelligence techniques become more important and successful compared to deterministic mathematical and physical methods. This text won't discuss the techniques developed to solve the piano movers problem, since these things are a bit out of the scope of the text. Some excellent introductions to the set of solutions can be found in [17] and [26].

11.2.10 Approximate methods vs. accurate and complete methods

It is often preferred to get a solution fast, rather than waiting much longer for a solution that is more exact or more optimal. This implies that many methods work with approximate, low-resolution models of the robot and its environment. For example, all objects are represented by rectangles or spheres. Such approximations inevitably lead to incomplete solutions from time to time, e.g., when planning the motion of a mobile robot amongst obstacles, an approximate method could possibly not find a solution because it underestimates the available free space.

11.3 Simple interpolation

This Section describes some of the simplest (hence most common) techniques to interpolate a motion from an initial pose to a final pose with zero velocity at both ends of the motion. The simplicity is mainly due to the fact that one *decomposes* the, in general, six-dimensional problem in a set of independent one- or two-dimensional subproblems, using straight line segments with easily calculatable time profiles. If one wants to pass through or near a given set of via-points (without stopping!), the straight lines are *blended* together with smooth curves (Sect. 11.3.2). The presented interpolation techniques are used in Cartesian space as well as in joint space, since one just interpolates between *coordinates* without taking into account the geometric properties of the motion represented by these coordinates. This means that the presented procedures are also applicable to mobile robots, to make them start, stop, or change speed in a "smooth" way.

11.3.1 Point-to-point

This Section gives three methods to interpolate a one-dimensional motion, e.g., from a joint angle q_0 to a joint angle q_T. T is the total time available to execute the motion. The result of the interpolation algorithms are the numbers $q(t)$, that give the desired value of the joint angle at time $t, 0 \leqslant t \leqslant T$.

Trapezoidal velocity profile The simplest approach, [53, 56], starts from q_0 with zero velocity; it then applies a constant acceleration $\ddot{q}$ until a maximum velocity $\dot{q}^{max}$ is reached; the motion continues with this velocity $\dot{q}^{max}$ until the deceleration phase starts, which applies a constant deceleration $-\ddot{q}$. The resulting velocity curve has the form of a symmetric trapezium (Fig. 11.1). Possibly the trajectory is so short that the maximum velocity is not reached; in this case, the trapezium degenerates into an isosceles triangle.

The major advantage of this technique is its simplicity; its major disadvantage is that it generates jumps in the acceleration, which can cause oscillatory behaviour of the robot.

Polynomial profile In order to avoid the acceleration jumps of the trapezoidal velocity profile, one can apply continuity constraints at the boundaries of the motion: the velocity, acceleration, jerk... have to vanish at the boundaries. For example, requiring continuous jerk imposes *four* constraints at each of the boundaries, so that at least a polynomial of the *seventh degree* has to be used. The drawback of this approach is that the form of the joint angle trajectory is not predictable: it follows from the constraints. If one requires more freedom in choosing the trajectory (e.g., by imposing intermediate points) the degree of the polynomial has to be increased, and hence

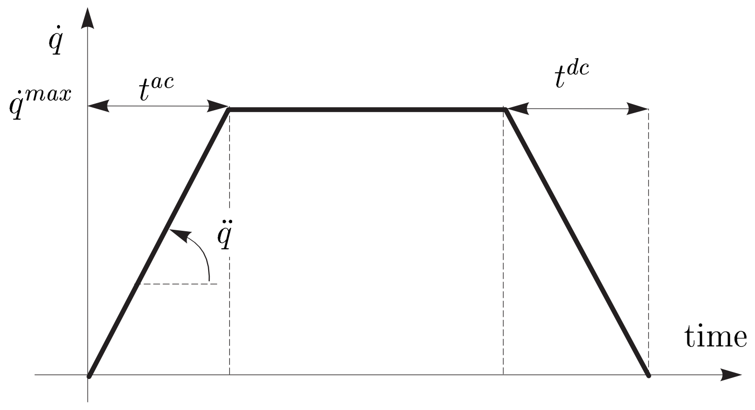


Figure 11.1: Trapezoidal velocity profile for joint interpolation. t^{ac} and t^{dc} denote the time intervals during which the joint is linearly accelerated, respectively decelerated.

the oscillatory behaviour can increase too. Also, one can not have a priori knowledge about what the maximum velocity or, more importantly, the maximum acceleration will be during the motion.

Continuous jerk profile Another approach to avoid the acceleration jumps of the trapezoidal velocity profile starts from the observation that smooth motions require continuity in the jerk of the trajectory. Hence, the motion is subdivided into segments with jerk profiles that start and stop with zero jerk, and that are continuous in the jerk. One possible choice is to use halves of sinusoids, [51]; another possibility could be to use trapezoidal jerk profiles. The major problem to be solved in these approaches is how to connect the available elementary jerk profiles in order to achieve a desired motion.

These methods allow to impose maximum acceleration and deceleration on the motion (see [51] for more details). This is an interesting feature, since acceleration is (more or less) proportional to the torque needed to perform the acceleration (cf. the Newton-Euler equations!), and hence one could take into account the acceleration bounds imposed by the dynamics of the robot and the characteristics of the actuators. Again, however, the final shape of the joint angle trajectory is not fully predictable.

11.3.2 Via-points—Blending

If one does not want the robot to stop in the via-points, one needs a way to smoothly connect two subsequent segments. The most common approach here (applicable to both joint space and Cartesian space interpolation) is as follows, [41, 50, 56]:

1. One uses one of the previous methods to generate a smooth start, and to accelerate to a given maximum velocity.

2. One continues with this constant velocity, i.e., a straight line in joint space or Cartesian space.

3. When one comes “close” to the next via-point, one applies an acceleration that makes a smooth transition from the current straight line to the straight line between the next via-point and the one after that. For example, by finding a parabolic segment tangent to both straight lines.

4. One smoothly decelerates the robot to a zero velocity at the end point.

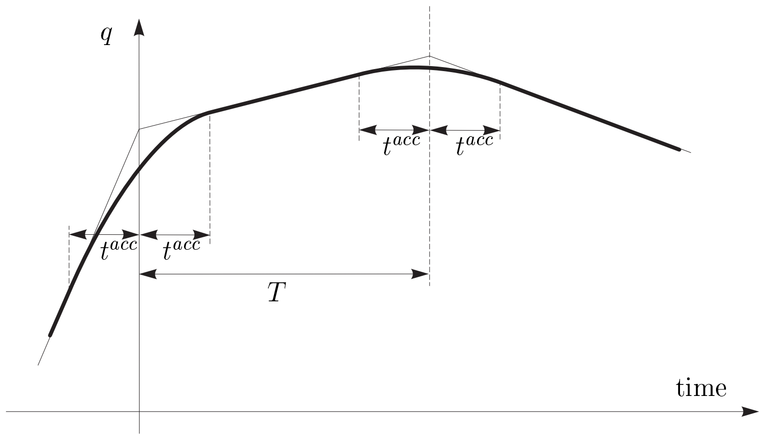


Figure 11.2: Blending of straight line segments in joint space. t^{acc} is the window during which the acceleration to, or deceleration from, a straight line segment takes place. T is the total time of the motion between two consecutive via-points.

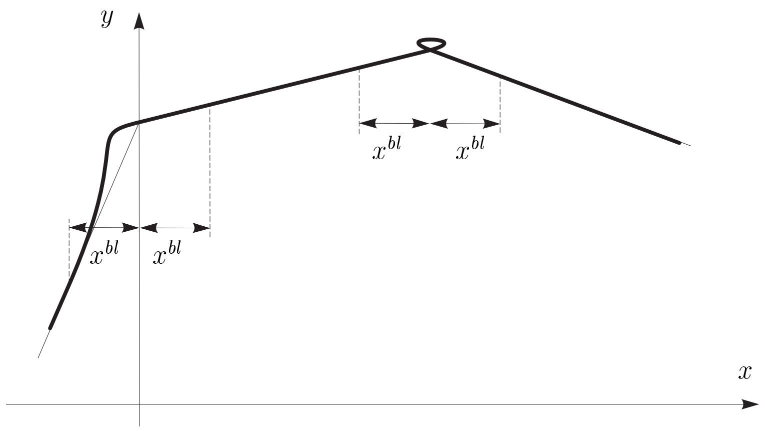


Figure 11.3: Blending (in Cartesian xy space) with more flexible user-specified parameters, [30]. x^{bl} is the blending preview window (in X direction!); cubic splines were used for the blending. The two transitions depicted in the figure have been blended with different goals in mind: in the first transition, the accurate tracking of the first line segment is given up for accurate tracking of the second line segment; the second and third line segment have to be followed accurately (for example, in order to apply or remove material to a workpiece), hence the transition between both cannot be done instantaneously and so the blending requires a "curl."

Figure 11.2 illustrates the concepts; the "acceleration window" t^{acc} is a user-defined parameter. Extensions of this method, [30, 51], use (i) non-symmetric transition windows and more flexible blending parameters, in Cartesian as well as in joint space (Fig. 11.3), (ii) non-linear segments, or (iii) continuous jerk profiles with non-zero velocities at the via-points (and hence, of course, also guaranteeing continuity in velocity and acceleration).

11.3.3 Tracking of small Cartesian errors

Sensor-based tracking is probably the simplest case of full Cartesian motion planning when the goal position and orientation are not specified beforehand: the robot is sensing its environment (e.g., by means of a camera, a force sensor, ultrasonic sensors, etc.) and tries to follow a particular feature in this environment (e.g., a visual marker, a desired contact force, or a given distance to the environment) even when this feature is *moving*. Hence, the tracking problem is mainly reduced to a continuous sequence of (i) sensing the new position and orientation of the feature with respect to the robot (see Chap. 12), and (ii) generating a tracking velocity that must reduce the "tracking error" between robot and feature. By the nature of this problem, the tracking error can be considered to be small, i.e., in general it can be modelled as an infinitesimal displacement twist $\mathbf{t}_\Delta$. The simplest method to generate a tracking velocity, or twist $\mathbf{t}$, is to divide the tracking error displacement twist $\mathbf{t}_\Delta$ by the sampling time T^s of the sensor system, and to use the robot's inverse velocity kinematics to generate corrective joint velocities, [55]:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}\mathbf{t} = \boldsymbol{J}^{-1}\frac{\mathbf{t}_\Delta}{nT^s}. \tag{11.1}$$

This equation means that the estimated pose error $\mathbf{t}_\Delta$ will be brought to zero in a period corresponding to n sample times T^s. Of course, this direct method is subjected to all "excitations" that a nervous or slow sensing system can introduce. From a control-theoretic point of view, it might be wiser to filter (or "smooth") this raw tracking velocity. Possible smoothing approaches are given in [8] and [30].

For mobile robots, a similar approach is often used, even though the "inverse velocity kinematics" are not defined. The method followed in this case is to have the user *define* an "inverse Jacobian" mapping from the Cartesian position and heading error to the velocities of the mobile robot's wheels, [16, 48]. This 2×3 inverse Jacobian depends, in general, on a number of arbitrary choices, i.e., the user has to make a particular trade-off between decreasing errors in X position, in Y position, and in heading.

11.3.4 Screw motion paths

Chasles's Theorem (Chap. 3) suggests another simple method to transform Cartesian pose "errors" into Cartesian twists, [42, 55], but now for *finite* pose errors: in the context of motion planning, this theorem says that any pose error can be annihilated by a certain translation along the screw axis, combined by a rotation about that screw axis. (Recall that rotation and translation commute *only* on the screw axis!) Hence, a possible screw motion trajectory is generated by applying one of the above-mentioned one-dimensional interpolators to the two geometric parameters that determine the pose error: the error distance along the screw axis, and the equivalent rotation angle about the screw axis.

The drawback of this method is that the resulting Cartesian trajectory is not very intuitive and hence hardly predictable. Especially for serial and parallel robots, this can often lead to collisions with either the environment or with the links of the robot itself.

11.3.5 Decoupled position and orientation interpolation

In order to avoid the problem mentioned in the previous paragraph, people have been looking for trajectories in which translation and rotation are separated in intuitive, predictable ways:

Translation. A chosen reference point on the robot can always be moved along a straight line from its initial to its final position. Possible collisions can be avoided by using multiple straight line segments around the obstacle. Hence, the linear interpolation techniques used in joint space are straightforwardly generalised to the translational motion planning for the robot's end-effector. The result is independent of the mathematical representation used to describe positions.

Rotation. Angular motion, however, requires a more careful treatment, due to the specific mathematical problems of the different possible representations (Euler angles, rotation matrix, unit quaternions, etc.) and the fact that orientations do not have a Euclidean metric (i.e., following "straight lines" in orientation space can make the robot end-effector perform very counterintuitive trajectories).

The following paragraphs give an overview of some of the problems of *orientation* interpolation, as well as methods for their solution. Combining these separate translation and rotation interpolators results *always* in trajectories that depend on the choice of reference point on the robot! Nevertheless, these results can be very satisfactory, since they are often quite intuitive.

Interpolation between line directions

From Section 4.3, we know that the shortest distance between two directions (i.e., between two *intersecting* lines) is the distance along the great arc through both directions and with its centre at their point of intersection. Hence, interpolation is straightforward: use one of the presented one-dimensional interpolators on the angle between both directions.

Interpolation between frame orientations

From Section 5.4.5, we know that the shortest distance between two frames (represented, for example, by two rotation matrices $\boldsymbol{R}^1$ and $\boldsymbol{R}^2$) is the equivalent angle of rotation of the relative orientation $\boldsymbol{R} = \left(\boldsymbol{R}^1\right)^{-1}\boldsymbol{R}^2$. Again, the classical one-dimensional joint space interpolators can be applied to this angle. Other approaches often interpolate each of the three Euler angles. However, this has several drawbacks:

1. This interpolation gives singularity problems: at the singularities of the Euler angle representation, infinite angle rates can occur, [1].
2. The generated trajectory depends on the chosen set of Euler angles.
3. The generated motion is often very counterintuitive.

Decoupled orientation interpolation

Richard Paul [41] developed an orientation interpolation technique that is very intuitive for most industrial manipulators: the change in orientation for the 3R wrist of a wrist-partitioned robot is decoupled in

1. A change in the first two wrist joint angles. This corresponds to the interpolation of two directions: the initial and final direction of the last joint axis. The previous Section gave an intuitive solution for this.
2. A change of the last joint angle, i.e., a rotation of the tool about the direction of the last joint axis.

The resulting trajectory is rather straightforward to predict intuitively: the tool will move along a great circle (combined of course with the translational motion of the end-effector) while at the same time rotating about the last joint axis, which coincides with the normal direction to the sphere on which this great circle lies. A similar "rotation minimizing motion" is described in [18].

11.4 Paths with geometric constraints—Potential fields

Many of the methods discussed in this Chapter are "free space" methods: it is tacitly assumed that the robot will not collide with the environment (or with itself) during the execution of the planned trajectory. So, other techniques had to be developed to cope with obstacles in the robot's workspace. Probably the most popular method is based on so-called *(artificial) potential fields*, [23, 24, 36, 45, 37, 43, 46]. The basic ideas behind this approach are:

1. Each obstacle to be avoided by the robot exerts an (artificial) *repelling* force on the robot.
2. The goal to be reached by the robot exerts an (equally artificial) *attractive* force on the robot.
3. These repelling and attractive forces are assumed to be deriveable from a *conservative potential field*, i.e., the change in the potential energy of the robot in the artificial potential field is proportional to the force exerted on the robot.
4. The motion of the robot is found by following the potential field "downstream."
5. The advantages of using potential functions as force generators are: (i) the influence of separate potential functions can be *added*, and (ii) they are straightforwardly integrated as external forces into the "Jacobian transpose" (Sect. 7.11), the Newton-Euler (Sect. **??**), and the Euler-Lagrange frameworks (Sect. 5.3).

The only limits on the choice of potential function are the user's creativity and the computer's computational power. This means that a plethora of different applications exist. The most attractive advantages of potential fields are their implementation simplicity and their intuitivity. Their common disadvantage are the inevitability of *local minima*, i.e., places in which the robot "gets stuck" because the forces generated by different real and/or artificial sources cancel each other. It is very difficult to avoid local minima, especially if several potentials are combined. One often-used solution to cope with them is to slightly "disturb" or "shake" the robot so that it can escape from the local minimum, [3].

Most practical implementations of potential fields use a *discretization* of the robot's space of possible positions and orientations, in order to limit the computational complexity of the problem. More about this in Section 11.8.

11.5 Optimization of the timing along a given path

Motion planning that tries to minimize the time required for a *given* geometrical path involves rather heavy mathematics (optimization of cost functions that rely on the explicit knowledge of the robot's dynamics) which is outside of the scope of this introductory text, [4, 19, 47]. However, some important and intuitively clear results are worth mentioning:

1. During the complete time-optimal motion, at least one of the joint motors works at maximum torque.
2. The motion planning then boils down to (i) finding when to switch the sign of the torque in the maximally loaded motor (this is called *bang-bang* control) and (ii) using the inverse dynamics of the robot to calculate the torques in all motors.
3. These time-optimal motions give the user no control over the frequencies that appear in the motion set-point signals. Hence, the chances increase to excite some natural frequencies of the robot system.

11.6 Smooth paths

Different smoothness criterions exist, [57]:

1. The integral of the norm of the instantaneous translational velocity of a reference point on the robot. This gives the shortest path.
2. Similar integrals of the norm of the acceleration and the jerk. As said before, acceleration and/or jerk are a computationally tractable trade-off between complexity and optimality.
3. The integral of the instantaneous power needed to drive the robot. This gives the minimum energy path.

The integrals to be minimized are of the following form:

$$I = \int_i^f \langle g(t), g(t) \rangle \, dt. \tag{11.2}$$

"i" and "f" are the initial and final positions of the robot, and $g(t)$ is one of the above-mentioned (time-dependent) functions. The brackets "$<,>$" denote an appropriate metric on *SE(3)*. What is "appropriate" depends on the task and on the user's preferences, [39, 40]. The reason to use the *integral* of the acceleration's norm has been discussed in the introduction to this Chapter too: smoothness alone is not sufficient to avoid oscillatory behaviour, one also has to keep the *magnitude* of the generated excitations as low as possible. Solving the minimisation problems of Eq. (11.2) involves techniques from the calculus of variations (cf. the Euler-Lagrange approach discussed in the Chapter on dynamics) plus differential geometry techniques to calculate the scalar product in all points on the trajectory. This text will not go into more details about the underlying mathematics, but some results are again worth mentioning, [57]:

1. No analytical solutions exist for the general problem, only numerical approximations.
2. In the minimum distance trajectory, the reference point chosen on the robot moves along a straight line, and the body rotates about an axis with fixed spatial direction. This looks a lot like the screw motion trajectories discussed above, but in the present case the translational and rotational directions do not necessarily coincide.
3. In the minimum acceleration and minimum jerk trajectories, the robot moves through exactly the same positions as in the minimum distance trajectory, but the *timings* along the path are different.

11.7 Mobile robot path planning

The following paragraphs discuss motion planning methods developed specifically for nonholonomic mobile robots in a planar environment *without obstacles*. As for serial and parallel manipulators, the user input consists of start and end positions and orientations, possibly extended with a number of via-points. The most important characteristics of a mobile robot in the context of motion planning are: (i) the problem is three-dimensional: two positions and one orientation degree of freedom; (ii) a mobile robot is a nonholonomic system, (iii) a limit exists on the maximum curvature of the paths it can follow, and (iv) basically, it has two basic motion operators (STEER and ROTATE). The methods described below are more or less ordered according to increasing complexity.

In this Section, the robot is assumed to move with *constant speed*. This is not a severe limitation, since most practical automatic vehicles indeed do move with more or less constant velocity. Moreover, many of the trajectory generation approaches of the previous Sections can be used to specify the transitions between regimes with different speeds. One important consequence of a constant velocity is that the *acceleration* of the mobile robot corresponds to v^2/r, with r the (instantaneous) radius of curvature of the robot's path, and v its (instantaneous)

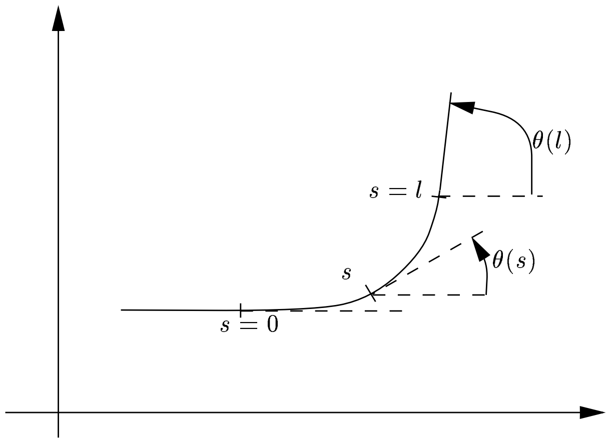


Figure 11.4: Clothoid of length l connecting two straight lines.

linear velocity. A constant velocity v then implies that the acceleration is proportional to the *curvature* $\kappa = 1/r$ of the mobile robot's path. Hence, the optimal trajectories defined above in terms of a minimal integral of the acceleration's norm translate into minimal integrals of the curvature's norm. Note that (i) curvature is a *scalar* quantity, and (ii) the robot's path is *fully determined* if the curvature of the path is known as a function of travelled distance.

11.7.1 Straight lines joined by circular arcs

Dubins [10] (forward driving only) and Reeds and Shepp [44] (forward *and* backward driving) proved the following facts:

Fact-to-Remember 65 (Shortest paths)
The shortest geometric paths for non-holonomic mobile robots are sequences of straight lines connected by circular arcs corresponding to the minimal radius of curvature allowed by the robot's kinematics. If backward driving is possible, the straight lines can also be linked by cusps, i.e., points where the robot changes from forward driving to backward driving or vice versa.

In practice, the shortest path is also *time-optimal* since driving is achieved with constant (hence maximum) speed. The major drawback of straight lines linked by circular arcs is that these trajectories produce *discontinuities in the curvature* of the path, hence discontinuities in the torques required from the wheel actuators, which increases the chance of slippage.

11.7.2 Segments joined by clothoids

Clothoids (or *Cornu* spirals) or *Euler spiral*, [12, 15, 20, 21, 28, 33, 49], are curves whose curvature increases or decreases *linearly* with arc length. (Their properties were described by the Swiss mathematician Jacob (James, Jacques) Bernoulli (1654–1705), [49, 7]; Euler (he again!) found a mathematical description; Cornu enters the scene in 1876 because he was the first to accurately draw them; and the Italian mathematician Ernesto Cesàro (1859–1906) coined the name “clothoid” in 1886; clothoids were popular at the end of the 19th century, for planning curves in railway tracks, [2].) Hence

$$\kappa(s) = \frac{d\theta(s)}{ds} = \beta s, \tag{11.3}$$

where $\kappa(s)$ is the curvature at arc length s along the clothoid, $\theta(s)$ is the direction of the tangent along the clothoid at the point with arc length s, and β is the linear proportionality constant (Fig. 11.4). For a curve with length l, and spanning a total orientation difference α, integration from $s = 0$ to $s = l$ gives

$$\theta(s) = 2\alpha \frac{s^2}{l^2}. \tag{11.4}$$

This result can also be found from the minimisation of a cost functional as in Eq. (11.2). Clothoid curves don't have analytical *position* solutions (they do have closed-form *tangent direction* solutions, Eq. (11.4)), so numerical approximations are required. Moreover, joining two *arbitrary* segments yields multiple *configurations*, [12, 20].

11.7.3 Segments joined by cubic spirals

Circular arcs minimize the sum of the squared acceleration along the arc, but have discontinuities at the transition points with other segments. Hence, smoother paths including the transition points will be found by minimizing the integral of the squared jerk while allowing arbitrary initial and final curvatures. These paths are called *cubic spirals*, [20]. As was the case for clothoids, cubic spirals have an analytical solution for the tangent direction:

$$\theta(s) = 3\alpha \frac{s^2}{l^2} - 2\alpha \frac{s^3}{l^3}. \tag{11.5}$$

Compared to circular arcs, cubic spirals have a larger turning radius, and are longer. Compared to clothoids, they are shorter but have a larger maximum curvature. Cubic spirals have the same problem when connecting two arbitrary segments: no closed-form solution is known, only numerical approaches work, and multiple configurations appear.

11.7.4 Elastica under tension

One of the optimisation criterions discussed in the previous Sections is the minimal integral of the norm of the acceleration; which, for mobile robots, reduces to

$$\min \int_0^l \kappa^2(s)\, ds. \tag{11.6}$$

This criterion takes only curvature information into account, and not the *length* of the curve. Criterion (11.6) is easily adapted to include a length minimisation too:

$$\min \int_0^l \left(\kappa^2(s) + \sigma\right)\, ds. \tag{11.7}$$

σ is the *tension* of the curve. A larger σ gives a solution with a shorter length, but a higher overall curvature. σ can be interpreted as a force applied at both ends of the curve, tangential to the curve: the higher this force, the shorter the curve, if this curve is considered to be an *elastic* rod that can store (elastic) potential energy. Hence,

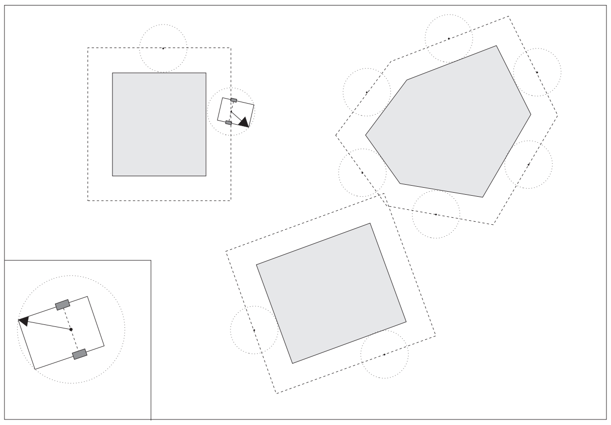


Figure 11.5: Growing of configuration space obstacles by the dimensions of the robot. For simplicity reasons, a circular omnidirectional robot is considered. (Figure courtesy of J. Vandorpe.)

these curves have been named *elasticas*, [5, 11, 31]. The snakes which are so common in the vision literature are just special cases of the elasticas.

Applying an Euler-Lagrange solution approach to the integral in Eq. (11.7) gives the following differential equation:

$$\frac{d\kappa(s)}{ds^2} + \frac{1}{2}\kappa^3(s) - \frac{1}{2}\sigma\kappa(s) = 0. \tag{11.8}$$

This differential equation has a “closed-form” solution in the form of *elliptic functions*, [5, 27]. (Elliptic functions were defined by the same Prussian mathematician Karl Gustav Jacob Jacobi (1804–1851) to whom also the matrix of first partial derivatives is named; this Jacobian matrix has been used extensively in the chapters on robot kinematics.) However, for arbitrary initial and final conditions, numerical procedures are needed, giving multiple configurations, just as for clothoids or cubic spirals.

11.7.5 Harmonic functions

Harmonic functions are the most complex but also most powerful model-based approaches to path planning. They are solutions to the Lagrange equation, that models the heath flow between a hot source and a cold sink, or the water flowing down from a source in the “mountains.” The current postion of the robot is the “source,” the goal position is the “sink.” Obstacles correspond to high temperature/altitude, so that they are naturally avoided. The drawback of solving the Lagrange equation in real time becomes already difficult for moderately complex environments. On the other hand, it can be proven to have no local minima (i.e., water will always find the sea).

11.8 Configuration space

The notion of *generalised coordinates* has already been used before in this text. The space of all generalised coordinates representing position and orientation is called the *configuration space*, [32]. For a mobile robot on a planar surface the configuration space is three-dimensional, i.e., SE(2). If no obstacles are present in the robot’s environment, the configuration space has coordinates in all of SE(2). When obstacles fill some regions in configuration space, the coordinates in SE(2) where the robot would collide with the obstacles are marked. In fact, the obstacles fill the region corresponding to their own space, which is “grown” by the dimensions of

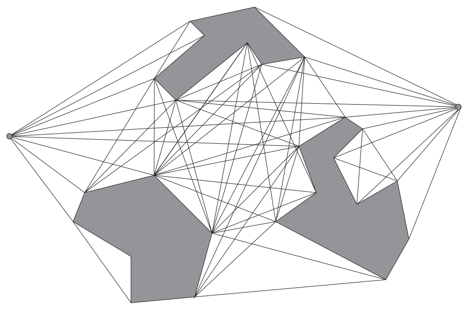


Figure 11.6: Graph of straight-line paths from start to goal in configuration space with obstacles. (Figure courtesy of J. Vandorpe.)

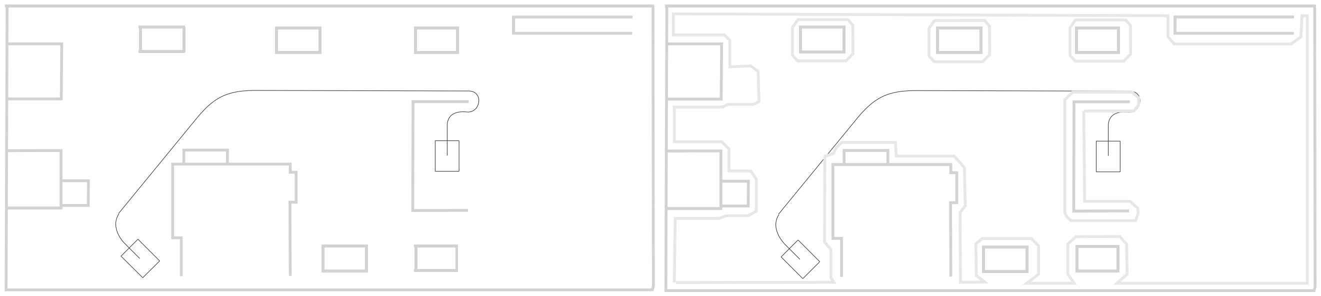


Figure 11.7: Simple nominal motion plan in map of PMA lab (left). This path results from a wavefront propagation procedure in the configuration space with grown obstacles (right). (Figure courtesy of J. Vandorpe.)

the robot (Fig. 11.5). In the simplest approximation, one doesn't take the *orientation* into account; hence the obstacle in configuration space is constructed by translating the outline of the robot around the contour of the obstacle. Figure 11.7 shows the result of a simple nominal motion plan (straight lines and circular arcs) in a map of the PMA lab, with and without grown obstacles.

One can divide the configuration space in "cells" that are either free or occupied. Motion planning in its simplest (i.e., holonomic) form then reduces to *graph searching*: find paths connecting the free cells of, respectively, the initial and desired final poses of the robot (Fig. 11.6). Hence, most of the well-known graph search algorithms can be applied to mobile robot motion planning. One of the best-known is the $A*$ algorithm, [14, 38, 54]. This is a *breadth-first* algorithm that decides on its next move by considering the sum of (i) the cost of the path already travelled, and (ii) the expected cost of the path to the goal. Selecting this last cost function is of course the difficult and creative part of the solution.

Most often the configuration space is *discretised*, i.e., only points on a *grid* are taken into account in the search procedures. This grid can be regular, or more detailed in regions where higher accuracy and resolution are required. The choice of grid resolution is a trade-off between computational effort and optimality of the solution. Another reason to use discretisation is that, for example, the potential in each grid point, resulting from all

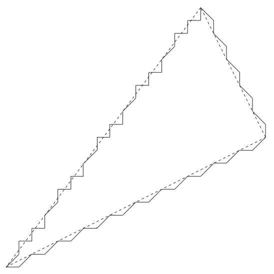


Figure 11.8: Each object is approximated by a sequence of straight lines with a user-specified length and a discretised orientation (four choices in the case depicted here). (Figure courtesy of J. Vandorpe.)

artificial potential fields in the environment, can be calculated off-line.

An alternative to the $A*$ algorithm is the *wavefront algorithm*, also called the *bush-fire algorithm*: one starts from the initial position of the robot in its configuration space, and one propagates one free cell further (propagation could mean different things: potential field, distance, cost, etc.); from this set of newly reached cells, one iterates the propagation to the next neighbouring free cell. (The physics behind this approach is *Huyghens principle*, which Christiaan Huyghens first used as a model for light rays.) One stops the iteration as soon as the goal position is reached.

Hybrid procedures The plans generated by the procedures above consist most often of straight lines in configuration space. This kind of paths do not satisfy the nonholonomicity constraints of most mobile robots. Hence, some *post-processing* of the paths is required. One interesting approach in this respect combines the advantages of configuration space planning (tractable speed for specified resolution) with the advantages of potential fields (flexibility; influences of different sources are easily combined): the motion plan generated by a configuration space planner (such as depicted in Fig. 11.7) is made into an elastic curve (*elastic band*, [43], or *snake*, [22]) that can be deformed on line in order to adapt the path to changes in the environment and obstacle outlines found from sensor data. The deformation ability of the snake is determined by the artificial dynamic properties of the curve (stiffness and mass) and of the environment space (damping generated by considering the space filled with a viscous fluid). The actual form of the trajectory then results from numerically solved dynamic algorithms (Euler-Lagrange type), [36, 43]. Incorporating a dynamics algorithm into the motion generation ensures that the on-line trajectory planner will not react too nervously to the inevitable sudden changes generated by sensor-based map building procedures.

11.8.1 Example of full nonholonomic planning

The off-line configuration space planners work mostly with (i) polyhedral approximations of all objects in the world, and (ii) connected straight lines as approximations to the real robot path. The nonholonomicity constraints of most mobile robots are only taken into account by a couple of more advanced (and hence computationally much more expensive) planners. Recently, a very fast, but discrete (hence suboptimal) full nonholonomic motion planner for a car-like mobile robot has been developed, [52]. It has efficient, approximate methods in each of the two classical planning steps:

1. *Configuration space construction.* This is a very heavy task, in the three-dimensional space SE(2) of the robot's position and orientation. A discretised configuration space is used, in which each object is first approximated by a sequence of small straight lines with a user-specified length and a discretised orientation (Fig. 11.8). Then each line segment in this approximated model is matched with the model of the robot (Fig. 11.9), which

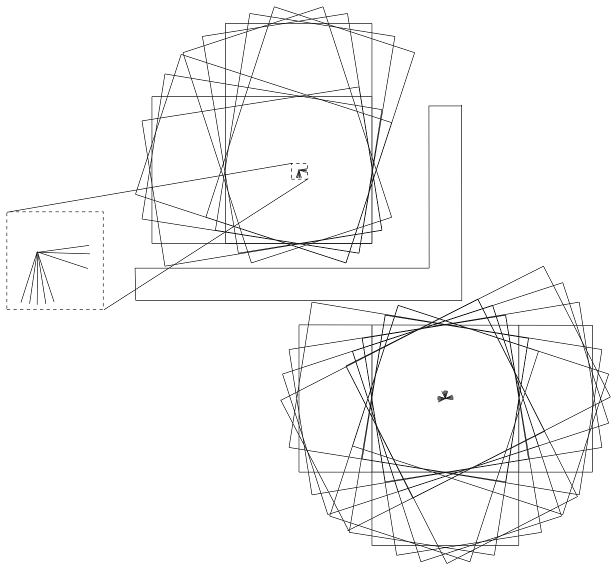


Figure 11.9: Typical step in the construction of the discretised free configuration space ("template") for a rectangular robot with an L-shaped obstacle. (Figure courtesy of J. Vandorpe.)

results in a set of allowed orientations for the mobile robot in each grid point of the configuration space. This matching can be done very fast by applying an off-line calculated *template* corresponding to each discretised line segment: the template consists of a set of discrete orientations in each grid point ("1" indicates free space, "0" indicates occupied space) in a user-specified area around the line segment. The template matching can be implemented by "inclusive or" operations which any computer can process very efficiently: start with a fully reachable configuration space (all discrete cells contain a "1"), and then apply an inclusive or with the template placed at each discretised line segment in the configuration space (i.e., wherever a "1" appears in the configuration space, and a "0" in the template, the configuration space "1" is overwritten by a "0").

2. *Path construction.* The construction of a feasible path in the free space generated in the previous step is also performed by a discrete template: the possible nonholonomic motions of the mobile robot during a small, user-specified time interval can be calculated off-line, and stored in a discrete template as depicted in Figure 11.10. These templates are applied in a *wavefront* procedure starting from the initial robot pose, until the desired end pose is reached. In each intermediate grid point one memorizes the predecessor templates that have been applied to this grid point most distantly in time and corresponding to each of the possible discrete orientations, such that the construction of the fastest path is a very straightforward backchaining in the lists of predecessors.

The method described above can be tuned by the user to be either more accurate or more efficient, by adapting the resolution of the discretisations applied in the templates and in the configuration space grid. Figure 11.11 shows two planning results generated by this method.

A final remark to conclude this Section: due to computational limitations, most implemented planners work in 2D, but full 3D applications begin to become feasible on high-end workstations, e.g., [25, 36].

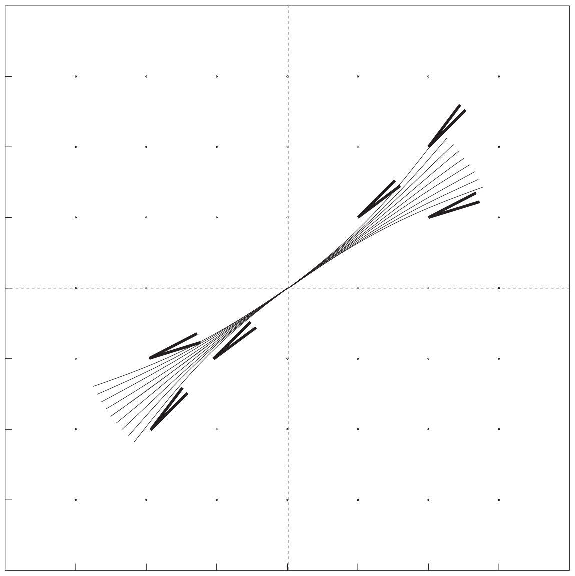


Figure 11.10: One step in wavefront procedure for mobile robot with limit on curvature radius. The continuous curves are paths with discretised turning radius; the bold "V" shapes are the resulting discretised final poses (i.e., approximations to nearest grid point) that the robot can attain. (Figure courtesy of J. Vandorpe.)

11.9 Force-controlled motion specification

Whenever the environment of the robot is not very structured (i.e., the robot doesn't have an accurate a priori map of the objects), the use of sensors is required to continuously adapt the information about the environment geometry. Chapter 12 deals with the case of dynamic map building for a mobile robot using ultrasonic distance sensors and a laser range finder; in that case, the *global* planning procedures of the previous Sections can be used each time the environment map has been updated. This Section presents a different kind of adaptive, sensor-based motion specification, applicable to tasks in which the robot uses sensors that give *local* information about the environment's geometry; examples of this kind of sensors are: a wrist force sensor, infrared, inductive or capacitive (short) distance sensors, local laser scanners, etc. The characteristic of these sensor-controlled motions is that one wants the robot to follow (one or more) surface(s) in the environment while (i) maintaining a user-specified relative position and/or orientation with respect to these surfaces and without making contact (in the case of desired distance between robot and environment), or (ii) making contact but without generating excessive contact forces.

This Section treats the latter case only: contacts occur between robot and environment, and a wrist force sensor measures the resulting contact forces; tasks like these are often called *compliant motions*, or *constrained motions*, [9, 35]. The former case of non-contact compliant motions is very similar: desired distances can be transformed into artificial "contact" forces by introducing a virtual elastic element between the robot and the environment, with user-specified impedance properties.

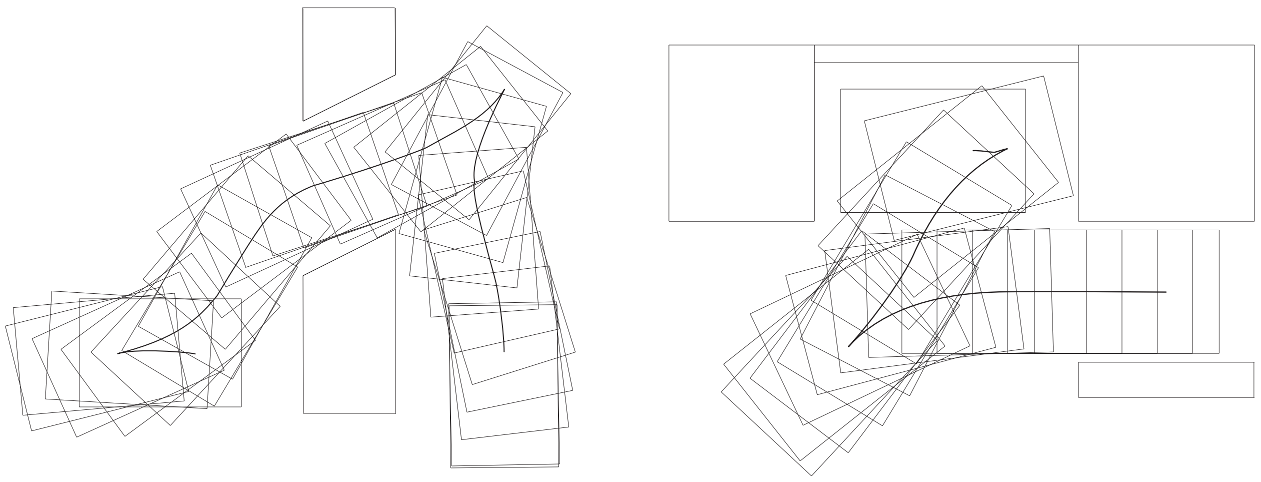


Figure 11.11: Two results of the discretised non-holonomic motion planner of Section 11.8.1. The total planning time is less than one second on a 150MHz personal computer. (Figure courtesy of J. Vandorpe.)

11.9.1 Physical and mathematical concepts

Constrained motion is completely and unambiguously described as follows [29, 34]: the rigid manipulated object must execute an instantaneous rigid body motion (twist $\mathbf{t} = [\boldsymbol{\omega}^T \boldsymbol{v}^T]^T$) that is *reciprocal* to all ideal (i.e., frictionless) reaction forces that the actual contact situation can possibly generate (wrenches $\mathbf{w} = [\boldsymbol{f}^T \boldsymbol{m}^T]^T$):

$$\boldsymbol{\omega}^T \boldsymbol{m} + \boldsymbol{v}^T \boldsymbol{f} = 0. \tag{11.9}$$

$\boldsymbol{\omega}$ is the angular velocity three-vector of the manipulated object; $\boldsymbol{v}$ is the linear velocity three-vector of a reference point on the rigid body, with respect to the origin of some reference frame; $\boldsymbol{f}$ is the linear force component (three-vector) of a possible ideal reaction force; $\boldsymbol{m}$ is the moment component, i.e., the sum of (i) a pure moment generated by the contact situation, and (ii) the moment generated by $\boldsymbol{f}$ applied at the reference point on the manipulated object, and with respect to the same origin as above.

The vanishing of the reciprocity condition Eq. (11.9) is a physical property (the motion $\mathbf{t}$ generates no power against the ideal reaction force $\mathbf{w}$) and is hence *invariant* under a change of world reference frame, a change of reference point on the manipulated object, and a change of physical units. It is universally valid for all contact situations between rigid bodies, however complex.

The vector space of all reciprocal twists in a given contact situation is called the *twist space* $\mathcal{T}$; the vector space of all ideal contact wrenches is the *wrench space* $\mathcal{W}$. The sum of the dimensions of $\mathcal{T}$ and $\mathcal{W}$ is always six; $\mathcal{T}$ and $\mathcal{W}$ are reciprocal vector spaces. Take the example of the cylindrical peg-in-hole task (Fig. 11.12): $\mathcal{T}$ is the two-dimensional space of translations along, and rotations about, the hole's axis (Z), and $\mathcal{W}$ is the four-dimensional space of moments about, and forces along, X and Y.

11.9.2 Motion specification

The principal motion specification tool is the so-called *Task Frame* (TF), or *compliance frame*: the TF is an orthogonal reference frame that is attached by the user to a characteristic geometric feature on either the robot or the environment. Examples of such characteristic features are: vertices, edges, planes, or kinematic joints (such as a hinge of a door). The TF has six programmable *task frame directions*: each of its three orthogonal

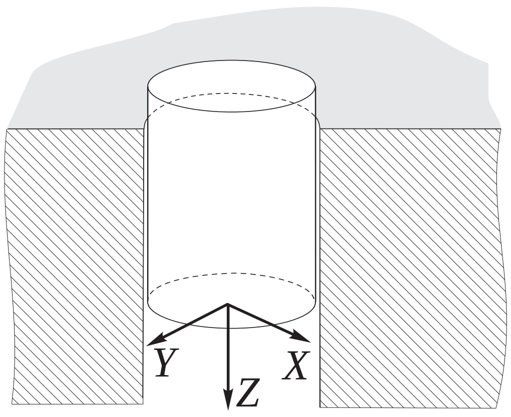


Figure 11.12: Insertion of a cylindrical peg.

axes is used once as an *axial vector* ([13], linear velocity or force), and once as a *polar vector* (angular velocity or moment). The task programmer places the TF in the contact configuration in such a way that (i) a basis for the twist vector space $\mathcal{T}$ consists of n_t ($n_t \leqslant 6$) TF directions (the so-called *position* or *velocity-controlled* directions), and (ii) a basis for the wrench vector space $\mathcal{W}$ consists of the remaining n_w ($n_w = 6 - n_t$) TF directions (the *force-controlled* directions). Figure 11.12 shows one possible TF for the peg-in-hole action: the origin is placed on the axis of the peg, with the Z axis parallel to the peg's axis. $\mathcal{T}$ is modelled by the axial and polar Z axes, and $\mathcal{W}$ by the axial and polar X and Y axes.

An elementary TF motion is specified by giving velocity set-points along the velocity-controlled TF directions, and force set-points along the force-controlled directions. These specifications (also called *artificial constraints*, [35]) must be compatible with the model of the constraint as described by the TF directions. The task continues until a *stop condition*, or *termination condition*, is fulfilled. This termination condition typically consists of a change in the contact situation. In the simplest case, this is indicated by one or more sensor measurements crossing a specified threshold (which is typically "zero"): gaining/loosing a contact corresponds to a sudden increase/decrease in the reaction force in a certain direction, as well as, dually, to the sudden decrease/increase of the velocity in this direction. The specification for the peg-in-hole task (Fig. 11.12) could be: (i) a desired translational velocity along the Z axis, (ii) no angular velocity about Z, (iii) no forces along, or moments about, the X and Y axes, and (iv) a force threshold crossing along the Z axis signals the end of the task. This force threshold should be greater than the expected friction during the action.

11.9.3 Tracking

In general, the geometry of the contact situation changes continuously during the motion. A TF motion should be able to adapt the position and orientation of the TF in such a way that the TF always remains compatible with the current contact, *and* keeps the *same* relative position and orientation with respect to the contact. This makes the specification simpler since, as seen from the TF, the contact situation remains time-invariant: (i) the force- and velocity-controlled TF directions do not change, and (ii) the task specification can use constant motion or force set-points. The adaptation of the TF's location is often called *tracking*. Tracking consists of corrective rotations about polar TF directions, and/or corrective translations along axial TF directions. Three types of tracking exist: (i) *no tracking*, i.e., the TF directions remain fixed to either the manipulated object (as in the peg-in-hole task); (ii) *model-based tracking*, i.e., the TF's update is directly derived from the constraint model as captured in the TF, and from the executed motion; and (iii) *sensor-based tracking*, i.e., the force and/or motion

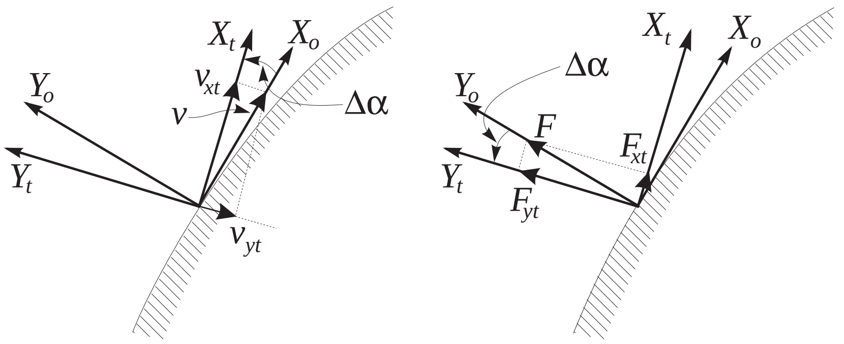


Figure 11.13: Tracking strategies, based "on velocities" and "on forces." The orientation error $\Delta\alpha$ is estimated by comparing the X and Y components of the measured force and velocity vectors. The index 0 indicates the real task frame, while the index t indicates the modelled task frame. v is the measured velocity; F is the measured force.

measurements are used to estimate how much the TF has moved away from a geometrically compatible position (this estimation procedure is called *identification* of the geometric TF uncertainties), and then these estimates are input to a *tracking* controller that generates corrective motions for the TF. The tracking is based on the instantaneous force and/or velocity measurements; the corresponding tracking strategies are called "tracking on forces," or "tracking on velocities," respectively. The identification of the orientation error $\Delta\alpha$ is illustrated by Fig. 11.13:

1. *Based on velocities*: the commanded velocity in the nominal task frame is changed by the force controller, in order to keep the manipulated object pushed against the environment. This requires a velocity component along the Y_t axis. $\Delta\alpha$ is estimated as $\Delta\alpha = \arctan(-v_{yt}/v_{xt}) \approx -v_{yt}/v_{xt}$.

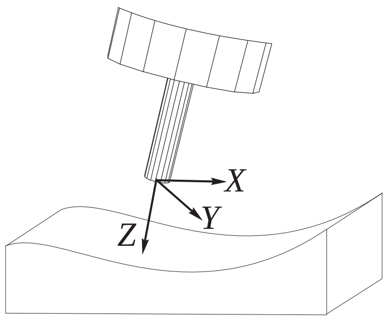
2. *Based on forces*: the nominal (ideal) reaction force lies along the Y_t axis, but a force component along X_t is measured because of the orientation error $\Delta\alpha$ between real and modelled task frame. This angle $\Delta\alpha$ is identified as $\Delta\alpha = \arctan(-F_{xt}/F_{yt}) \approx -F_{xt}/F_{yt}$.

The orientation error $\Delta\alpha$ is continuously fed back to the model, with certain tracking control dynamics. (This does not mean that the physical motion of the end-effector must follow this tracking.) Identification based on forces is disturbed by contact friction; identification based on velocities is disturbed by compliance of the robot, the manipulated object and the environment.

11.9.4 Some examples

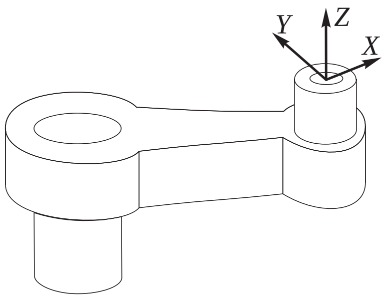
The following paragraphs give an overview of elementary force-controlled motions that can be specified by the TF formalism, [6]. Each motion is illustrated by a sketch of the contact situation, the "standard" location of the task frame, and a textual motion specification of the task. The semantics of the task specification are not formally defined; they should be clear from the context. The actions are more or less ordered according to the complexity of the underlying control. This complexity is mainly due to (i) increased flexibility in the specification, and (ii) increased uncertainty in the contact models. Remark that for most tasks the task specification is not unique.

Guarded approach (Fig. 11.14). This is the simplest action. The force "control" is only used to detect the transition between the no-contact and contact states: the termination condition of the task consists of detecting



```
move compliantly {
    with task frame directions
        xt: velocity 0 mm/sec
        yt: velocity 0 mm/sec
        zt: velocity $v$ mm/sec
        axt: velocity 0 rad/sec
        ayt: velocity 0 rad/sec
        azt: velocity 0 rad/sec
    } until zt force < -$f$ N
```

Figure 11.14: Guarded approach motion. The robot moves the manipulated object in free space, and stops when a contact force is felt. Remark the convention used in the textual specification for force measurements and set-points: the reaction force is directed from the environment into the manipulated object.



```
move compliantly {
    with task frame directions
        xt: force 0 N
        yt: velocity $v$ mm/sec
        zt: force 0 N
        axt: force 0 Nmm
        ayt: force 0 Nmm
        azt: force 0 Nmm
    } until time > $t$ sec
```

Figure 11.15: Turning a crank.

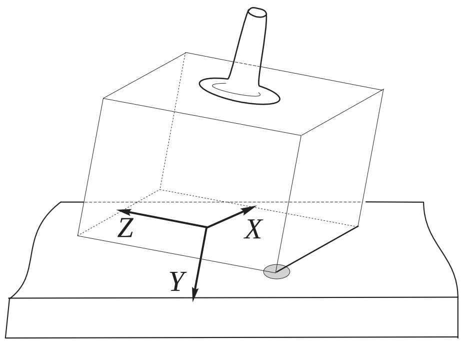
a (sufficiently non-zero) reaction force. The task frame is placed somewhere on the object, with, for example, the axial Z axis being the direction in which the robot must move in order to make contact.

Turning a crank (Fig. 11.15). The robot is connected to a revolute joint, whose rotation axis is not exactly known. However, many cranks have *two* revolute joints: one on the axis of the crank, another one at the crank's handle. This last joint allows the robot to change its orientation with respect to the task frame. This "redundancy" has two simple solution alternatives: (i) the end-effector orientation remains constant with respect to the task frame, and (ii) the end-effector orientation remains constant with respect to the absolute world.

Aligning a block with a surface (Fig. 11.16). In this task, force control serves two purposes: (i) it pushes the block against the surface with a certain force in order to ensure a contact; (ii) it regulates two reaction moments to zero, in order to align the contacting face of the block with the surface.

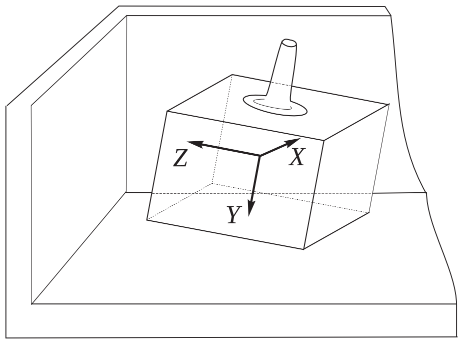
Placing a block in a corner (Fig. 11.17). This is equivalent to three of the previous tasks in parallel, i.e., one for each face of the block.

2D contour tracking, or edge following (Fig. 11.18). The task can be described as "follow an unknown and arbitrary two-dimensional contour in the XZ plane while moving at constant tangential speed and while applying a constant normal force; the orientation of the end-effector is to remain constant with respect to the world." The



```
move compliantly {
    with task frame directions
        xt: velocity 0 mm/sec
        yt: force -$f$ N
        zt: velocity 0 mm/sec
        axt: force 0 Nmm
        ayt: velocity 0 mm/sec
        azt: force 0 Nmm
} until time > $t$ sec
```

Figure 11.16: Aligning a block with a surface.

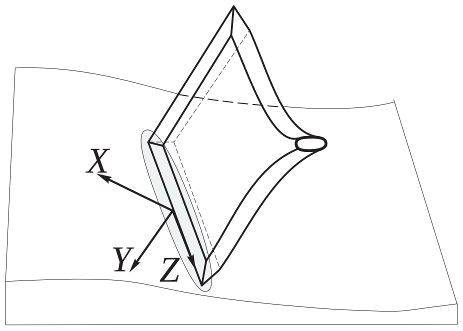


```
move compliantly {
    with task frame directions
        xt: force -$f_x$ N
        yt: force -$f_y$ N
        zt: force -$f_z$ N
        axt: force 0 Nmm
        ayt: force 0 Nmm
        azt: force 0 Nmm
} until time > $t$ sec
```

Figure 11.17: Placing a block in a corner.

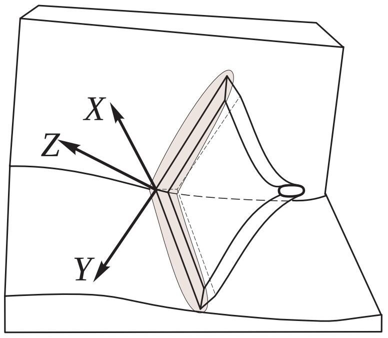
Y direction is to be the normal direction, while the tangential velocity is specified along the X direction. The task frame's orientation about the Z axis must continuously be adapted ("tracked") to the changing direction of the contact normal.

Following a 3D seam (Fig. 11.19). This task is also called *compatible seam tracking*, since the shape of the manipulated object is compatible with the shape of the seam. It is the combination of two *2D contour following*



```
move compliantly {
    with task frame directions
        xt: velocity $v$ mm/sec
        yt: force $f$ N
        zt: velocity 0 mm/sec
        axt: velocity 0 rad/sec
        ayt: velocity 0 rad/sec
        azt: track (on velocities)
} until until distance > $d$ mm
```

Figure 11.18: 2D contour folowing. The task frame's X axis is tangential to the contour; Y is the outward pointing normal.



```
move compliantly {
    with task frame directions
        xt: force -$f_x$ N
        yt: force -$f_y$ N
        zt: velocity $v$ mm/sec
        axt: track (on velocities)
        ayt: track (on velocities)
        azt: force 0 Nmm
} until distance > $d$ mm
```

Figure 11.19: Following a 3D seam.

tasks. The Z direction of the task frame is parallel to the tangent to the seam; the X and Y axes point into the two seam surfaces. The Z axis is velocity-controlled in translation, with the desired velocity as set-point, and force-controlled in orientation, with zero set-point. Desired forces are exerted along X and Y, and tracking is needed about both axes.

References for this Chapter

[1] M. H. Ang, Jr. and V. D. Tourassis. Singularities of Euler and Roll-Pitch-Yaw representations. *IEEE Transactions on Aerospace and Electronic Systems*, AES-23(3):317–324, 1987.

[2] R. C. Archibald. *American Mathematical Monthly*, 25:276–282, 1918.

[3] J. Barraquand and J.-C. Latombe. A Monte-Carlo algorithm for path planning with many degrees of freedom. In *IEEE Int. Conf. Robotics and Automation*, pages 1712–1717, 1990.

[4] J. E. Bobrow, S. Dubowsky, and J. S. Gibson. Time-optimal control of robotic manipulators along specified paths. *Int. J. Robotics Research*, 4(3):3–17, 1985.

[5] G. Brunnett and J. Wendt. A univariate method for plane elastic curves. Technical Report 262/94, Universität Kaiserslautern, Kaiserslautern, Germany, 1994.

[6] H. Bruyninckx and J. De Schutter. Specification of force-controlled actions in the “Task Frame Formalism”: A survey. *IEEE Trans. Rob. Automation*, 12(5):581–589, 1996.

[7] E. Cesáro. *Vorlesungen über natürliche Geometrie*. B. G. Teubner, Leipzig, Germany, 1901. Authorised German translation by Gerhard Kowalewski of “Lezioni di geometria intrinseca”.

[8] S. Chand and K. L. Doty. On-line polynomial trajectories for robot manipulators. *Int. J. Robotics Research*, 4(2):38–48, 1985.

[9] J. De Schutter and H. Van Brussel. Compliant robot motion I. A formalism for specifying compliant motion tasks. *Int. J. Robotics Research*, 7(4):3–17, 1988.

[10] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497–516, 1957.

[11] L. Euler. Methodus inveniendi lineas curvas maximi minimive proprietate gaudentes, sive solutio problematis isoperimetrici latissimo sensu accepti, additamentum II (1744). In C. Carathéodory, editor, *Opera omnia*, pages LII–LV, 298–308. Fussli, Zürich, Switzerland, 1952.

[12] A. Gentile, A. Messina, and A. Trentadue. Composed primitives in smooth local path planning for mobile robots. In *Proceedings 9th IFToMM*, pages 2302–2307, 1995.

[13] H. Goldstein. *Classical mechanics*. Addison-Wesley, 1980.

[14] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. on Systems, Science, and Cybernetics*, 4(2):100–107, 1968.

[15] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. AK Peters, 1993.

[16] G.-J. Huang. *World modelling, path planning, and navigation of an autonomous mobile robot*. PhD thesis, Katholieke Universiteit Leuven, Dept. Mechanical Engineering, Leuven, Belgium, 1994.

[17] Y. K. Hwang and N. Ahuja. Gross motion planning—A survey. *ACM Computing Surveys*, 24(3):219–291, 1992.

[18] B. Jüttler. *Rotation minimizing spherical motions*. In *Recent Advances in Robot Kinematics*, pages 413–422, 1998.

[19] M. E. Kahn and B. Roth. The near-minimum-time control of open-loop articulated kinematic chains. *Trans. ASME J. Dyn. Systems Meas. Control*, 93:164–172, 1971.

[20] Y. Kanayama and B. I. Hartman. Smooth local path planning for autonomous vehicles. In *IEEE Int. Conf. Robotics and Automation*, pages 1265–1270, 1989.

[21] Y. Kanayama and N. Miyake. Trajectory generation for mobile robots. In O. D. Faugeras and G. Giralt, editors, *Robotics Research: the Third International Symposium*, pages 333–340, 1986.

[22] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, pages 321–331, 1988.

[23] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Robotics Research*, 5(1):90–99, 1986.

[24] O. Khatib and J.-F. Le Maitre. Dynamic control of manipulators operating in a complex environment. In *Third CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators*, pages 267–282, 1980.

[25] Y. Koga and J.-C. Latombe. On multi-arm manipulation planning. In *IEEE Int. Conf. Robotics and Automation*, pages 945–952, 1994.

[26] J. C. Latombe. *Robot motion planning*. Kluwer, 1991.

[27] D. F. Lawden. *Elliptic functions and applications*. Springer, 1989.

[28] J. D. Lawrence. *A catalog of special plane curves*. Dover, 1972.

[29] H. Lipkin and J. Duffy. Hybrid twist and wrench control for a robotic manipulator. *Trans. ASME J. Mech. Transm. Automation Design*, 110:138–144, 1988.

[30] J. Lloyd and V. Hayward. Trajectory generation for sensor-driven and time-varying tasks. *Int. J. Robotics Research*, 12(4):380–393, 1993.

[31] A. E. H. Love. *A treatise on the mathematical theory of elasticity*. Dover, 1944.

[32] T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

[33] H. Makino. Clothoidal interpolation—a new tool for high-speed continuous path control. In *Annals of the CIRP*, volume 37/1, pages 25–28, 1988.

[34] M. Mason and K. Salisbury. *Robot Hands and the Mechanics of Manipulation*. MIT Press, 1985.

[35] M. T. Mason. Compliance and force control for computer controlled manipulators. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-11(6):418–432, 1981.

[36] A. McLean and S. Cameron. The virtual springs method: Path planning and collision avoidance for redundant manipulators. *Int. J. Robotics Research*, 15(4):300–319, 1996.

[37] F. Miyazaki, S. Arimoto, M. Takegaki, and Y. Maeda. Sensory feedback based on the artificial potential for robot manipulators. In *Proc. 9th IFAC*, pages 2381–2386, 1984.

[38] N. J. Nilsson. *Principles of artificial intelligence*. Tioga, 1980.

[39] F. C. Park. Distance metrics on the rigid-body motions with applications to mechanism design. *Trans. ASME J. Mech. Design*, 117:48–54, 1995.

[40] F. C. Park. Optimal robot design and differential geometry. *Trans. ASME J. Mech. Design*, 117:87–92, 1995.

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

[42] D. L. Pieper. *The kinematics of manipulators under computer control*. PhD thesis, Stanford University, Department of Mechanical Engineering, 1968.

[43] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *IEEE Int. Conf. Robotics and Automation*, volume 2, pages 802–807, 1993.

[44] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[45] M. Renaud. Contribution à l'étude de la modélisation des systèmes mécaniques articulés. Master's thesis, Université Paul Sabatier, Toulouse, 1976.

[46] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Trans. Rob. Automation*, 8(5):501–518, 1992.

[47] Z. Shiller and H.-H. Lu. Computation of path constrained time optimal motions with dynamic singularities. *Trans. ASME J. Dyn. Systems Meas. Control*, 114:34–40, 1992.

[48] K.-T. Song. *Planning and control of a mobile robot based on a ultrasonic sensor*. PhD thesis, Katholieke Universiteit Leuven, Dept. Mechanical Engineering, Leuven, Belgium, 1989.

[49] D. J. Struik. *Lectures on classical differential geometry.* Addison-Wesley, 1961.

[50] R. H. Taylor. Planning and execution of straight line manipulator trajectories. *IBM Journal of Research and Development*, 23(4):424–436, 1979.

[51] L. Van Aken. *Robot motions in free space: task specification and trajectory planning.* PhD thesis, Katholieke Universiteit Leuven, Dept. Werktuigkunde, 1987.

[52] J. Vandorpe. *Navigation techniques for the mobile robot LiAS.* PhD thesis, Katholieke Universiteit Leuven, Dept. Werktuigkunde, 1997.

[53] K. J. Waldron. Geometrically based manipulator rate control algorithms. *Mechanism and Machine Theory*, 17(6):379–385, 1982.

[54] C. W. Warren. Fast path planning using modified a^* method. In *IEEE Int. Conf. Robotics and Automation*, volume 2, pages 662–667, 1993.

[55] D. E. Whitney. The mathematics of coordinated control of prosthetic arms and manipulators. *Trans. ASME J. Dyn. Systems Meas. Control*, 94:303–309, 1972.

[56] W. A. Wolovich. *Robotics: basic analysis and design.* Holt, Rinehart and Winston, 1986.

[57] M. Žefran and V. Kumar. On the generation of smooth three-dimensional rigid body motions. In *Computational Kinematics '95*, 1995. (Not in proceedings.)

Chapter 12

Intelligent sensor processing

12.1 Introduction

A necessary (but certainly not sufficient!) condition for a robot to act intelligently, and to achieve autonomy and robustness, is to use sensors so that it can find out how its environment looks like, and how it reacts with it. Humans excel in using their sensorial capabilities, and, in the robotics context of object manipulation, they do so especially with respect to their visual and tactile sensing capabilities. The state of the art in artificial sensing comes not even close to this human sensing performance. The raw sensing itself is sometimes more accurate than the human counterparts, e.g., distance or force measurement. But the major part of sensing is the *intelligent interpretation* of the sensed data, i.e., making the transition from pure sensor *data* to *information* that is relevant to the task the robot is currently performing, *and* that helps in completing this task. (Note, in this context, the very significant difference between "information" and "data"!) At the very lowest (hardware) level, sensor data processing consists of two steps:

1. *Filtering.* The raw sensor signals are processed in order to eliminate, as much as possible, the influence of measurement noise, and the frequencies in the signals that are not important for the current task of the robot. For example, if one wants a mobile robot to drive autonomously on a highway, it is not important to measure the changes in the direction of the road at a level of, let's say, less than one meter.

2. *Transformation.* Most often, the measurements that come out of the filter have to be interpreted in a setting that is relevant to the current task. This requires a transformation from the "sensor frame(s)" to the "task frame(s)" in which the robot task is specified. Transformation can consist of purely geometric operations only (e.g., transforming the distance measured by one sonar sensor on the rim of a mobile robot to the distance as "seen" from the reference frame somewhere else on the robot) as well as of combinations of data (so-called *sensor fusion*) from different sensors. For example, to determine the direction of a wall next to the mobile robot, one combines the distances to this wall as measured by sensors at different spots on the rim of the robot.

This text doesn't consider filtering and transformation as "intelligent" sensor processing. On the other hand, a typical example of what "intelligent" sensing really means is *map building*: a mobile robot drives around in an unknown environment, using ultrasound or vision sensors, or a force-controlled robot moves its tool over an unknown (set of) objects. The robots detect "landmarks" in that environment, i.e., features of the environment that are particularly easy to detect with their sensors. For example: corners for ultrasound or force sensors; specific textures for vision; particular contact situations or transitions for force sensors. Fully autonomous map building is not yet state of the art, but for the easiest applications (i.e., mobile robot navigation) one has come quite close (see Sect. 12.5)!

This text considers map building as "high-level" robot control. It is supported by lower-level control techniques:

1. *Low-level control.* The *state* of the model is estimated from the measurements, and its evolution over time is *tracked.* The Bayesian tool par excellence for this low-level control is the *Kalman Filter*, Sect. 12.3.1.

2. *Medium-level control.* In all but the simplest tasks, the robot passes through different stages in the task, each with its own particular model. Hence, detecting which model is the correct one, and when a switch of model is required, is the goal of the medium-level robot control. Once a model is chosen, the medium-level controller must estimate where in the model the current sensor readings fit. The corresponding Bayesian tool is the *Hidden Markov Model*, Sect. 12.4.

The Bayesian tool at the high-level control is the *EM algorithm*, Sect. 12.5. "Bayesian tools" are characterized by the following properties:

1. *Model = Bayes network.* The robot system and its environment are physical objects, whose "interactions" are detected by the sensors. The network models how the *visible parameters* of the model (i.e., the ones that the sensors can detect) are influenced by the *hidden parameters* (i.e., the parameters that determine in which sub-task the robot system is, and what the current interaction is). The goal of the Bayesian tools is to estimate these hidden parameters.

2. *Uncertainty = Probability distribution.* The above-mentioned influences between hidden and visible paramters are modelled as *conditional probability distributions.* General probability distributions can be quite complicated to work with in computer algorithms. Hence, some sets of "practical" families of distributions have been developed in the past century. The most popular family is the *exponential family.* It requires only a finite (and often small) number of parameters to represent a wide range of probability density functions. Well-known members of the exponential family are, [1, 27]: the normal distribution ("Gaussian"), the discrete distributions, and the *mixture* distributions (i.e., weighted sums of Gaussians).

3. *Bayes' rule.*

4. *Maximum Likelihood Estimation (MLE).* The likelihood is an important part of Bayes' rule. It represents how well a particular model explains the measured data. Hence, the "best" estimate of the model parameters is the one that maximizes the likelihood. This is the most popular technique, although one also often uses *Maximum A Posteriori* (MAP) estimation: not just the likelihood is maximized, but the product of the likelihood and the prior, which gives the a posteriori distribution of the parameters.

Fact-to-Remember 66 (Basic ideas of this Chapter)
This Chapter presents model-based "intelligent" sensor processing techniques, based on Bayesian probability theory. Three typical techniques are covered: the Kalman Filter for state estimation and tracking, the Hidden Markov Model for map navigation, and the EM algorithm for map building.

12.2 Robot sensors

The sensors used in robots are roughly divided into proprioceptive and exteroceptive sensors. Most industrial robots use only the former, while one or more of the latter are indispensible for intelligent and autonomous robots:

Proprioceptive sensors. These measure *internal* properties of the robot, i.e., motion of the joints, and/or forces exerted at the joints. The following paragraphs give the most common sensors, without details about the physical properties underlying their working. (See, e.g., [6] for a more thorough discussion on these physical properties.)

1. Encoders and resolvers: these sensors measure the relative angle between two consecutive links in a robot. Encoders come in two types: (i) the cheaper *incremental encoders*, that can only measure *changes* in the angles, and (ii) *absolute encoders*, that measure the angle with respect to a fixed reference on the shaft. If the transmission from motor to robot joint has a significant gear ratio (as is the case with most industrial robots) then an encoder placed on the motor shaft gives more accurate joint angle measurements than an encoder placed directly on the joint axis.

2. Tachos: these sensors produce a voltage that is proportional to the *speed* of a motor or joint axis. This information could be (and in practice often is) constructed also by differentiation of the position signals produced by encoders; however, these signals are most often in binary format, so that numerical differentiation gives inaccurate results for low speeds or high sampling frequencies.

   Tachos are needed if one wants to implement *velocity control* on the robot joints.

3. Current or torque sensors: these sensors measure the current in the motor armatures, or, more directly, the torque that works on the joint axis. This information is necessary if one wants to implement *torque control* on the robot joints.

4. Accelerometers: these measure the acceleration (in one or more spatial directions) at the point of the robot to which they are attached. Note that (i) accelerometers give inaccurate results if the acceleration is small, and (ii) angular acceleration is much more difficult to measure than linear acceleration.

Exteroceptive sensors. These measure interaction properties of the robot with its environment:

1. Force/torque sensors (or "force sensors" for short). These measure one to six components of the wrench (i.e., linear force as well as moment) exerted on the sensor. Physically, force sensors are most often *deformation sensors*: they measure their own deformation under the influence of the external force, and then multiply this deformation by their stiffness matrix. If these deformations are small, one uses *strain gauges*; for large deformations, optical measurements are more appropriate.

   Force sensors give very *local* and *indirect* information, i.e., only the contact points with the environment generate signals, and it is difficult to find out from the resulting force and moment where and how the contacts occur.

2. Distance sensors. These come in many types and prices, with strongly varying characteristics. Some of the more popular ones are:

   (a) Ultrasonic sensors. These are cheap, but give rather inaccurate results, due to the fact that (i) they use rather wide beams (> 10 degrees), and (ii) the propagation of sound waves through the air is influenced by turbulences of the air and by the temperature of the air. US sensors measure distances from about half a meter to about fifteen meter.

   (b) Infrared sensors. These measure the distance to an object by comparing the emitted and reflected intensities of an infrared beam. It is obvious that these sensors are influenced by the surface properties of the environment. Infrared sensors work from a couple of millimeters to a couple of centimeters.

(c) Laser range sensors. These work on either the *time-of-flight* principle (i.e., multiplying the speed of light by the time between emission of a pulse and reception of the echo) or on a *phase shift* detection between the outgoing and incoming signal. Laser range sensors are meant for measurement over rather large distances (from a couple of meters to many hundreds of meters.)

(d) Linear Variable Differential Transformers. An LVDT can very accurately (order of magnitude $5\mu m$) measure distances from about half a millimeter to a couple of centimeters.

3. Gyroscopes. To measure the changes in the *absolute* orientation of a robot.
4. Cameras. CCD cameras (Charged Coupled Device) are most frequently used. Their output is an array of *pixels* ("picture elements), containing gray or color values. Typical sizes of images are 640×512. Camera images have the big advantage that they give very *global* information about a scene; their big disadvantage is that it is so very, very hard to write computer code that "sees" the same things in the image as a human.
5. Tactile arrays: these are "artificial skin" sensors, consisting of an array of pressure-sensitive points. Their output is comparable to the image of a CCD camera, but with much lower resolution. Tactile arrays are, for example, used to detect contact points on the fingers of a robotic hand.

12.3 State estimation and tracking

If one knows what physical process the sensor is observing, and what the underlying physical properties are, one can make a so-called *estimator* (or *observer*) i.e., a *digital* "filter" that uses a *mathematical model* of the physical process, and extracts from the sensor signal the parameters of this model that most closely correspond to the expected physical behaviour of the observed system. Note that this digital observer should use the output from the analog low-pass filter, instead of the raw sensor data, unless these raw data are of a very high quality. In robotics, observers are often used as state estimators: they estimate the position, velocity, and/or acceleration of the robot's joints, or of objects in the "field of view" of the sensors, [10, 15, 22, 25, 26].

Typically, a digital filter works with a fixed *sampling period*, i.e., every T^s units of time a new measurement is taken and processed. For robotic purposes, it is also often necessary that the filter works *on line*, i.e., the robot needs the output of the filter as soon as possible after the new measurement is taken. Such an on-line filter is also called a *recursive filter.* The basic components of *linear* filters or estimators are (Fig. 12.1):

The state of the observed system. As said before, an estimator works with a mathematical model of the physical process it observes. This means that it represents the current *state* of the process by a set of real numbers, that each represent the last estimate of one particular parameter in the model. All these numbers are assembled in what is called the *state vector* $\hat{\boldsymbol{x}}(k)$. The index k refers to the instant in time (measured in discrete numbers of sample periods T^s) at which the state vector is valid; the hat "ˆ" denotes the *estimated* state vector, while $\boldsymbol{x}(k)$ is the real, unknown state. The filter is recursive in the sense that $\hat{\boldsymbol{x}}(k)$ is calculated from all previous states $\hat{\boldsymbol{x}}(j), j \leqslant k$, and all measurements $\boldsymbol{z}(j), j \leqslant k$, up to the time instant k. Often, the state $\hat{\boldsymbol{x}}(k)$ contains all information available about the process at instant k. (In this case, the system under observation is called a *Markov process.*) Hence, the new state estimate $\hat{\boldsymbol{x}}(k+1)$ is to be calculated from $\hat{\boldsymbol{x}}(k)$ and $\boldsymbol{z}(k)$ only.

Prediction of next state. If the estimator has a mathematical model $\boldsymbol{F}$ of the observed process, it can use this model to predict the next state:

$$\hat{\boldsymbol{x}}(k+1|k) = \boldsymbol{F}(k)\hat{\boldsymbol{x}}(k), \tag{12.1}$$

if the model is *linear*, or, more generally,

$$\hat{\boldsymbol{x}}(k+1|k) = \boldsymbol{F}(k)(\hat{\boldsymbol{x}}(k)), \tag{12.2}$$

if the model is *nonlinear.* $\hat{\boldsymbol{x}}(k+1|k)$ denotes the prediction of the state at instant $k+1$ if the estimate of the state at time instant k is known.

For example, if the state vector $\boldsymbol{x}$ contains (one dimension of) the position p and velocity v of an observed object, the state prediction function $\boldsymbol{F}$ (under the assumptions of a sample period of T^s, and constant velocity!) looks like:

$$\boldsymbol{x}(k+1) = \begin{pmatrix} 1 & T^s \\ 0 & 1 \end{pmatrix} \boldsymbol{x}(k). \tag{12.3}$$

Prediction of next measurement. Similarly, the value of the next measurement can be predicted from the model of the process, and a model $\boldsymbol{H}$ of the sensing action:

$$\hat{\boldsymbol{z}}(k+1|k) = \boldsymbol{H}(k)\hat{\boldsymbol{x}}(k+1|k), \quad \text{or} \quad \hat{\boldsymbol{z}}(k+1|k) = \boldsymbol{H}(k)(\hat{\boldsymbol{x}}(k+1|k)). \tag{12.4}$$

Innovation. If the model is completely correct, *and* if the estimated state corresponds exactly to the real state, *and* if there are no noise or other disturbance sources in the system, then the predicted measurement will correspond exactly to the new measurement. Of course, these three pre-conditions are never completely fulfilled in practice, so the predicted and real measurement will differ:

$$\boldsymbol{\nu}(k+1) = \boldsymbol{z}(k+1) - \hat{\boldsymbol{z}}(k+1|k), \tag{12.5}$$

where the vector $\boldsymbol{\nu}$ is called the *innovation*, [17], i.e., the new information that the latest measurement brings to the observer. Information is to be understood here in the sense that the innovation contains that part of the evolution of the observed system that the observer was not able to predict.

Correction. The predicted state $\hat{\boldsymbol{x}}(k+1|k)$ is now adapted to give a new state estimate $\hat{\boldsymbol{x}}(k+1)$ by means of a *feedback law* based on the latest innovation. Most often, this feedback is simply proportional:

$$\hat{\boldsymbol{x}}(k+1) = \hat{\boldsymbol{x}}(k+1|k) + \boldsymbol{W}(k+1)\boldsymbol{\nu}(k+1). \tag{12.6}$$

The matrix $\boldsymbol{W}(k)$ is called the *gain matrix.* In general, it can change from sample instant to sample instant, hence the index k. For a deterministic observer, this gain matrix is usually constant, and determined by classical algorithms such as pole placement, see e.g., [13].

12.3.1 Kalman filter

The previous Section gave an overview of the basics behind state estimators. In practice, many sensors are subjected to significant levels of "*noise.*" For example, ultrasonic sensors give a distance measurement that heavily depends on the turbulences in the air, as well as on the air temperature. Hence, the *distance* estimation is noisy. Moreover, the sensing beam of the US sensor is relatively wide, so that also the *direction* estimation is noisy. These are two examples of *measurement noise.* A second source of noise is often called *process noise*: this stems either from an intrinsically noisy physical process (for example Brownian motion of molecules in a liquid), or from a process whose evolution in time is not accurately known. In robotics, noisy processes are not very common, but "uncertain" (stochastic) processes are. For example, the distance between the wheels of a mobile

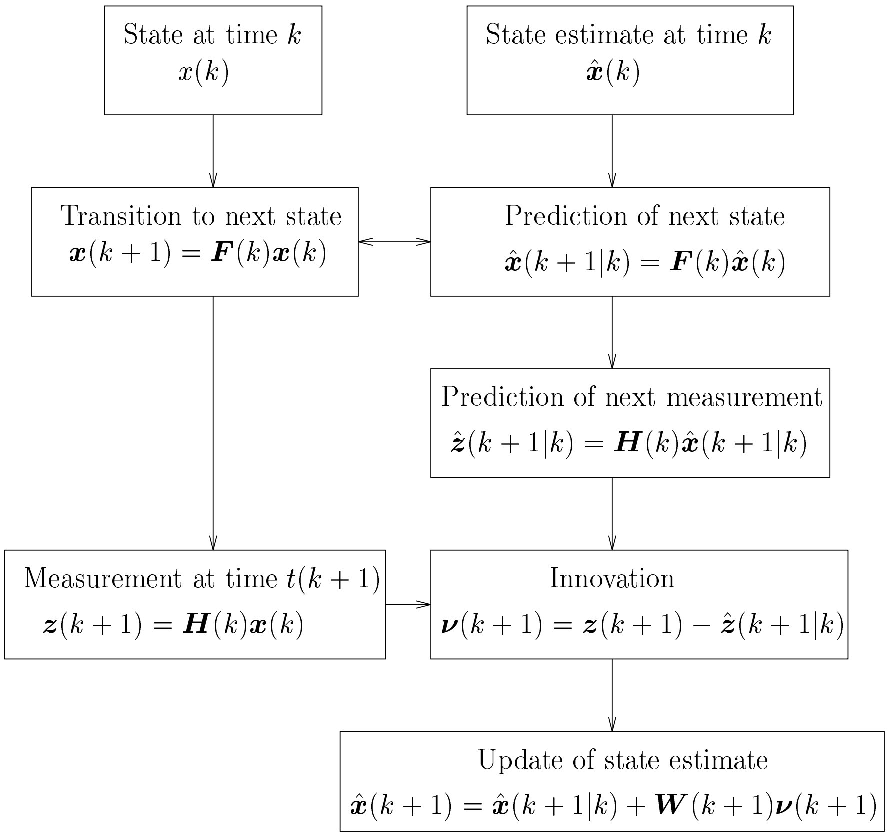

State at time k
$x(k)$
State estimate at time k
$\hat{\boldsymbol{x}}(k)$
Transition to next state
$\boldsymbol{x}(k+1) = \boldsymbol{F}(k)\boldsymbol{x}(k)$
Prediction of next state
$\hat{\boldsymbol{x}}(k+1|k) = \boldsymbol{F}(k)\hat{\boldsymbol{x}}(k)$
Prediction of next measurement
$\hat{\boldsymbol{z}}(k+1|k) = \boldsymbol{H}(k)\hat{\boldsymbol{x}}(k+1|k)$
Measurement at time $t(k+1)$
$\boldsymbol{z}(k+1) = \boldsymbol{H}(k)\boldsymbol{x}(k)$
Innovation
$\boldsymbol{\nu}(k+1) = \boldsymbol{z}(k+1) - \hat{\boldsymbol{z}}(k+1|k)$
Update of state estimate
$\hat{\boldsymbol{x}}(k+1) = \hat{\boldsymbol{x}}(k+1|k) + \boldsymbol{W}(k+1)\boldsymbol{\nu}(k+1)$

Figure 12.1: Digital estimator: overview of the method.

robot cannot be known exactly since rubber wheels change their contact points with the ground under changing loads and driving conditions.

One very popular recursive estimator is the *Kalman filter*. It takes into account process uncertainty as described above. The Kalman filter can be derived in many ways. One is as an *optimum linear least-squares filter*: it minimises the mean of the square of the error between the measurement and the prediction, [3, 14, 18, 19, 28, 29]. (In fact, it reformulates the classical least-squares solution to a curve fitting problem, as it was originally described by Gauss; the innovative ideas of Kalman were to find a *recursive* implementation.) Another approach to derive the Kalman Filter is to start with Bayes' rule: if the prior probability is a Gaussian, and the sensor uncertainty is also represented by Gaussians, then the posterior probability is also a Gaussian, [8, 21, 24, 33]. The Maximum Likelihood Estimation of a Gaussian is simple, since it coincides with its mean.

All these properties will not be proven here; the interested reader is referred to the given literature. The following paragraphs just give the extensions to the deterministic estimator described in the previous Section. These extensions (Fig. 12.2) involve the *covariances* of the states and the measurements. For example, the covariance of a state vector $\boldsymbol{x}(k)$ is the matrix of the expectation values of $(\boldsymbol{x}(k)-E(\boldsymbol{x}))(\boldsymbol{x}(k)-E(\boldsymbol{x}))^T$, ($E(\boldsymbol{x})$ is the expected value, i.e, the mean, of the state vector $\boldsymbol{x}$) and is a measure of how much certainty one has about the estimate $\hat{\boldsymbol{x}}(k)$ of the state $\boldsymbol{x}(k)$, given the uncertainty on the process and the measurements. Roughly speaking: the larger the covariance, the more uncertain the estimate. So, the estimation procedure of the previous Section is adapted as follows:

The state of the observed system. The current knowledge about the state $\hat{\boldsymbol{x}}(k)$ is uncertain, and this uncertainty is modelled by a covariance matrix $\boldsymbol{P}(k)$. At the start of the estimation process the user has to fill in some intial value for this covariance.

Prediction of next state. Also the state prediction $\hat{\boldsymbol{x}}(k+1|k)$ is subjected to uncertainty, stemming from the fact that the current state vector $\hat{\boldsymbol{x}}(k)$ is uncertain, as wel as, in general, the observed physical process itself. When the filter is running, the update of the state covariance according to the model is found as follows:[1]

$$\boldsymbol{P}(k+1|k) = \boldsymbol{F}(k)\boldsymbol{P}(k)\boldsymbol{F}^T(k) + \boldsymbol{Q}(k). \tag{12.7}$$

The covariance matrix $\boldsymbol{Q}(k)$ represents the *process uncertainty* at time instant k. The value of this covariance depends on the uncertainty characteristics of the process.

Prediction of next measurement—Innovation. Similarly for the prediction of the measurement: this prediction is subjected to a so-called *innovation covariance*, denoted by $\boldsymbol{S}(k)$:

$$\boldsymbol{S}(k+1) = \boldsymbol{H}(k)\boldsymbol{P}(k)\boldsymbol{H}^T(k) + \boldsymbol{R}(k). \tag{12.8}$$

The covariance matrix $\boldsymbol{R}(k)$ represents the *measurement noise* (or uncertainty) at time instant k. The value of this covariance depends on the noise characteristics of the sensors.

Correction. In order to obtain a minimum least-squares estimator, the gain matrix $\boldsymbol{W}(k+1)$ of Eq. (12.6) is calculated as

$$\boldsymbol{W}(k+1) = \boldsymbol{P}(k+1|k)\boldsymbol{H}^T(k+1)\boldsymbol{S}^{-1}(k+1). \tag{12.9}$$

We refer to the literature for a thorough derivation of this result. What is important in the context of this course is to realize that:

1. A large gain matrix means that much emphasis is put on the newly arrived measurement, Eq. (12.6).
2. The gain is large if (i) the state is inaccurately known (large covariance $\boldsymbol{P}$), (ii) the sensitivity of the observations to changes in the state is large (large $\boldsymbol{H}$), or (iii) the measurement uncertainty is low (small covariance $\boldsymbol{S}$), [8].
3. Contrary to the deterministic estimator, the Kalman filter has an *automatically adaptable gain*: the gain changes according to how uncertain the latest estimate was.
4. If a *constant* real state vector is estimated, the Kalman filter estimate converges to this real state vector. This means that the gain reduces to zero (the filter becomes very "stiff") and hence the filter will become very insensible to newly occurring changes in the state.

Finally, with this newly calculated gain, the state covariance is updated as follows:

$$\boldsymbol{P}(k+1) = \boldsymbol{P}(k+1|k) - \boldsymbol{W}(k+1)\boldsymbol{S}(k+1)\boldsymbol{W}^T(k+1). \tag{12.10}$$

The minus sign in this equation implies that the estimate of the state becomes less uncertain: this is what should be expected, since the estimator has taken into account new measurement information.

This Section ends with some practical facts about Kalman filters:

[1]We leave out the details of the derivation of this formula; just note that the update involves the process update mapping $\boldsymbol{F}$.

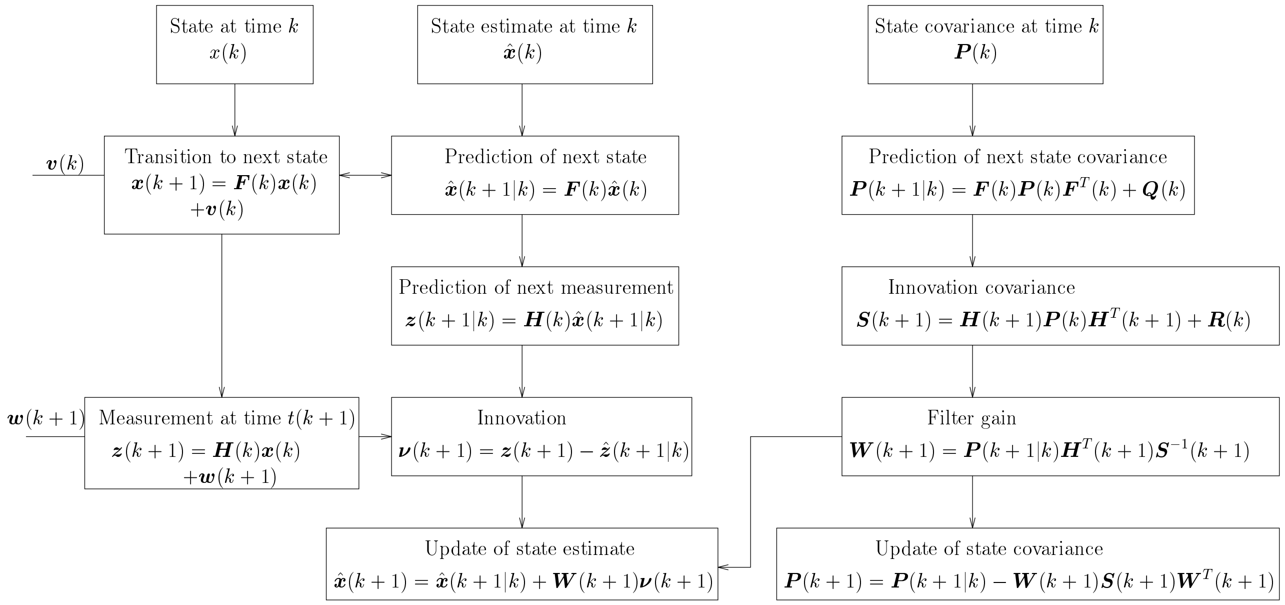

State at time k
$x(k)$
State estimate at time k
$\hat{\boldsymbol{x}}(k)$
State covariance at time k
$\boldsymbol{P}(k)$
$\boldsymbol{v}(k)$
Transition to next state
$\boldsymbol{x}(k+1) = \boldsymbol{F}(k)\boldsymbol{x}(k) + \boldsymbol{v}(k)$
Prediction of next state
$\hat{\boldsymbol{x}}(k+1|k) = \boldsymbol{F}(k)\hat{\boldsymbol{x}}(k)$
Prediction of next state covariance
$\boldsymbol{P}(k+1|k) = \boldsymbol{F}(k)\boldsymbol{P}(k)\boldsymbol{F}^T(k) + \boldsymbol{Q}(k)$
Prediction of next measurement
$\boldsymbol{z}(k+1|k) = \boldsymbol{H}(k)\hat{\boldsymbol{x}}(k+1|k)$
Innovation covariance
$\boldsymbol{S}(k+1) = \boldsymbol{H}(k+1)\boldsymbol{P}(k)\boldsymbol{H}^T(k+1) + \boldsymbol{R}(k)$
$\boldsymbol{w}(k+1)$
Measurement at time $t(k+1)$
$\boldsymbol{z}(k+1) = \boldsymbol{H}(k)\boldsymbol{x}(k) + \boldsymbol{w}(k+1)$
Innovation
$\boldsymbol{\nu}(k+1) = \boldsymbol{z}(k+1) - \hat{\boldsymbol{z}}(k+1|k)$
Filter gain
$\boldsymbol{W}(k+1) = \boldsymbol{P}(k+1|k)\boldsymbol{H}^T(k+1)\boldsymbol{S}^{-1}(k+1)$
Update of state estimate
$\hat{\boldsymbol{x}}(k+1) = \hat{\boldsymbol{x}}(k+1|k) + \boldsymbol{W}(k+1)\boldsymbol{\nu}(k+1)$
Update of state covariance
$\boldsymbol{P}(k+1) = \boldsymbol{P}(k+1|k) - \boldsymbol{W}(k+1)\boldsymbol{S}(k+1)\boldsymbol{W}^T(k+1)$

Figure 12.2: Kalman filter: overview of the method.

Forgetting factor/fading memory. In a very dynamic environment, it is often necessary to "forget" old data, since it doesn't give any information anymore about the current state of the system. Hence, extensions to the above-mentioned Kalman filter exist, that apply fading to old measurements and estimates.

Fudging. Even if the estimate of the Kalman filter gets better and better (and hence the uncertainty represented in the covariance matrix decreases) when collecting more sensing data, it might be interesting to avoid its covariance to drop below a certain threshold. If one does not do this, the filter will become very "rigid," in the sense that it is very sure about its estimate; this makes it very difficult to react to sudden changes in the observed system. Hence, if such sudden changes are expected, an artificial level of covariance (sometimes ironically called the "fudge factor") can be introduced, in order to keep the filter active and alert.

Statistical checks. The stochastic parameters int he Kalman filter are not only used to model the uncertainties in the measurements and the state estimates; they can also be used to check whether the models $\boldsymbol{F}$ and $\boldsymbol{H}$ are still valid representations of the observed system. If these are indeed valid models, the innovations $\boldsymbol{\nu}(k)$ should have a *white noise* statistical distribution; i.e., they should be completely random. However, if the model is not a faithful representation of the observed system, the statistical distribution of the innovations will exceed some user-defined, statistical *confidence level*. Most often, so-called χ^2 ("chi-square") tests are used, [3].

Extended (or modified) Kalman filter . If the measurement and/or state models $\boldsymbol{H}$ and $\boldsymbol{F}$ are *nonlinear*, one could still apply the general form of the estimator, as described above. The state and meaurement predictions then use the *linearized* parts of $\boldsymbol{F}$ and $\boldsymbol{H}$ around the current *estimated* state.

Sensor fusion. Intuitively speaking, it should be possible to "fuse" the data coming from different sensors, with different covariances, into one single "artificial sensor" with higher accuracy. For example, the mobile robot

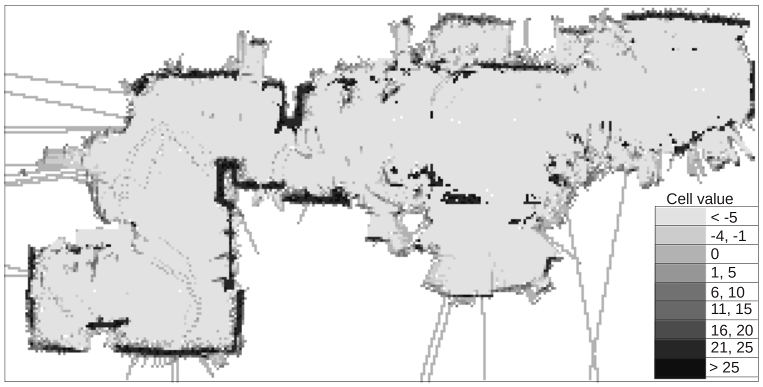


Figure 12.3: Raw data from laser range finder scan. (Figure courtesy of J. Vandorpe.)

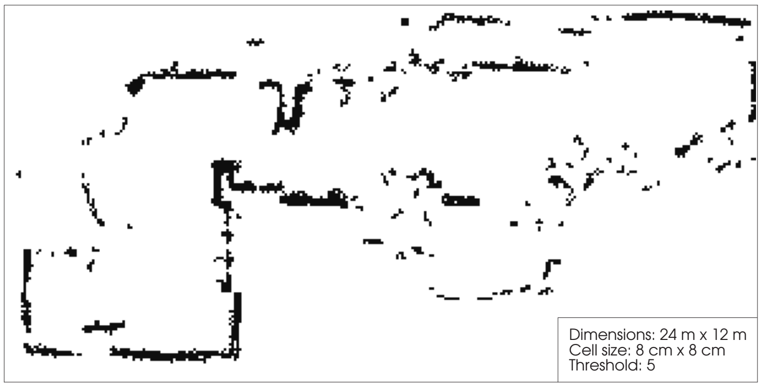


Figure 12.4: Figure 12.3 after thresholding. (Figure courtesy of J. Vandorpe.)

position and velocity estimates could be the result of such an integration between (i) the forward kinematics based on wheel encoder measurements, (ii) the orientation updates given by a gyroscope, and/or (iii) the meaurements of "landmarks" in the environment for which the absolute position is a priori known to the robot.

What is the basic principle of sensor fusion? Assume that n sensors give measurements $\boldsymbol{z}_i, i = 1, \dots, n$, with covariances $\boldsymbol{R}_i$, of the *same* observed feature. Then, the best estimate $\hat{\boldsymbol{z}}$ for the fused measurement would be, [12],

$$\hat{\boldsymbol{z}} = \left(\sum_{i=1}^{n} \boldsymbol{R}_i^{-1} \right)^{-1} \left(\sum_{i=1}^{n} \boldsymbol{R}_i^{-1} \boldsymbol{z}_i \right), \qquad \text{with covariance} \qquad \boldsymbol{R} = \left(\sum_{i=1}^{n} \boldsymbol{R}_i^{-1} \right)^{-1}. \tag{12.11}$$

12.3.2 Application: mobile robot position tracking

This Section presents the results from the implementation of some of the sensor signal processing techniques discussed in the previous Sections, combined with some of the path planning primitives described in Chapter 11. The target system is the mobile robot LiAs (Fig. 9.1), that uses ultrasonic and laser range sensors to measure

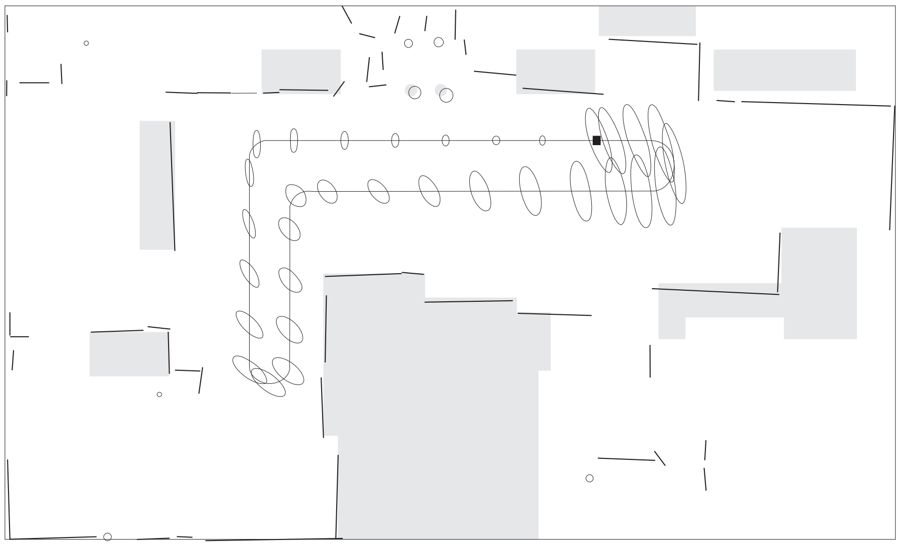


Figure 12.5: Result of dead reckoning navigation based on proprioceptive sensors only (i.e., encoders and gyroscope). The uncertainty increases along the path. The line segments sensed during the motion are matched in a least-squares sense with the a priori map of the room. (Figure courtesy of J. Vandorpe.)

the distance to obstacles, and its wheel encoders and gyroscope to perform the proprioceptive dead reckoning (Sect. 9.7.1).

Dead reckoning through sensor fusion

LiAS uses Kalman filter techniques to fuse the motion information generated by (i) its encoders, and (ii) its gyroscope. The (change in) orientation information coming from both sources is weighted according to the covariance on both signals. If, however, the gyroscope's resolution is far higher than the resolution from the encoders, it's easier (and better) not to fuse the "bad" data from the encoders with the "perfect" data from the gyroscope: just use the gyroscope data as only measurement.

Low-level filtering on obstacle detection

The ultrasonic sensor and the laser range scanner both give information about in which direction and at what distance obstacles are detected. Again, Kalman filter techniques fuse the information from both sources. Figure 12.3 shows the result, on a scan of one room in the PMA lab. The map is discretised in cells of a given length, and each cell contains a number representing the probability that an obstacle is present in this cell. This data is then thresholded, such that a "black-and-white" picture of the room remains: black means "obstacle," white means "free space" (Fig. 12.4). The threshold is a parameter that the user must tune.

The following step in the filtering is the fitting of simple geometric features (lines and circles) to the thresholded bitmap. Figure 12.5 shows the result of all the previous low-level filtering steps, collected during a closed trajectory of the mobile robot, and overlayed on a known a priori map. The Figure makes clear that the line fitting becomes less accurate in the regions where the robot has less accurate information about its own pose in the room. This uncertainty increases along the motion of the robot, since the dead reckoning alone generates inevitable drift.

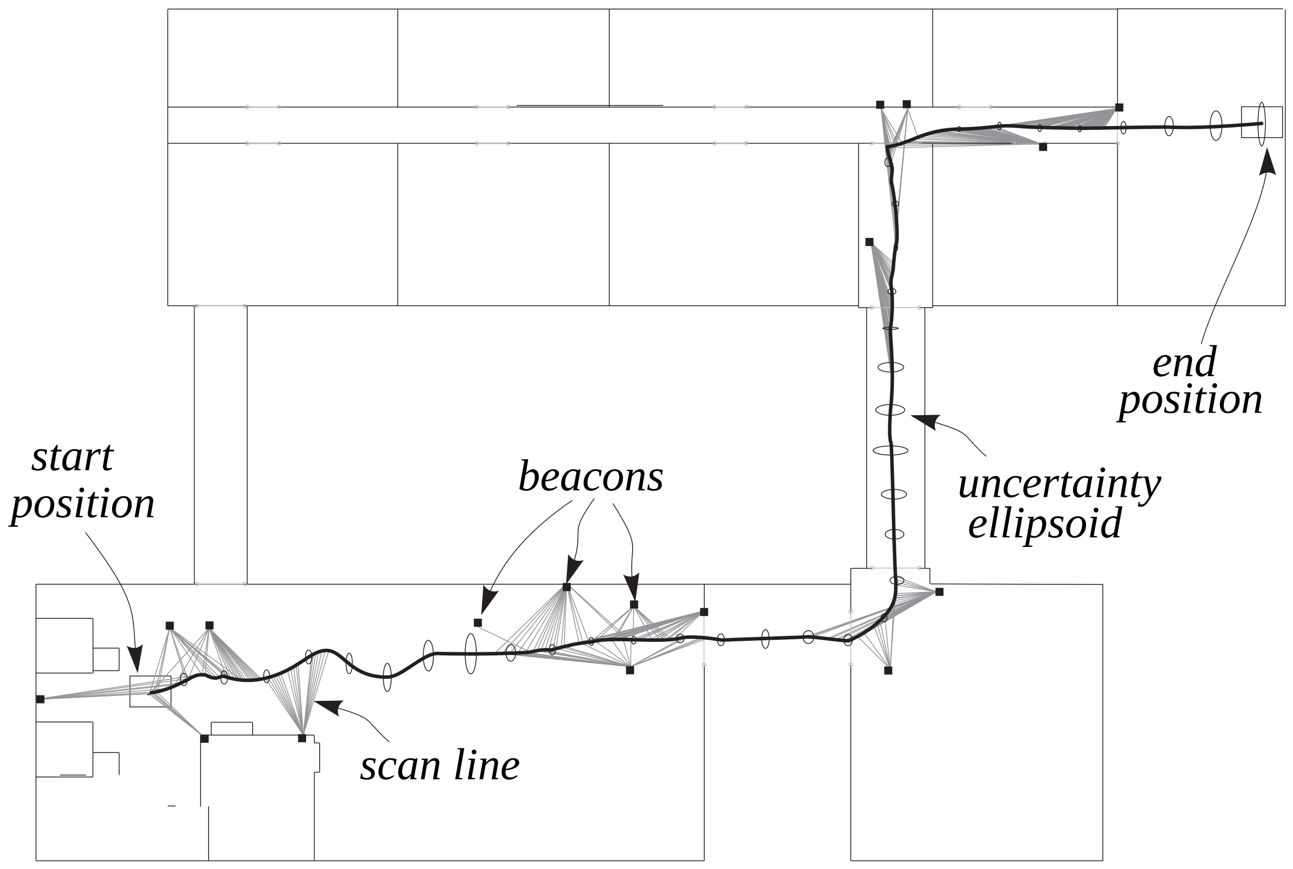


Figure 12.6: Result of navigation with recalibration based on fixed beacons. Whenever the robot is able to see beacons in its environment (and is able to match them with its a priori map) it can reduce its position uncertainty. This is reflected by the covariance ellipses shown in the picture. (Figure courtesy of J. Vandorpe.)

LiAS uses a second source of geometric information: at known positions in the room highly reflective strips are attached to the walls; the laser scanner can easily detect these strips, and hence has a number of directions in which it sees these beacons.

Matching with a priori map

The geometric features found by the low-level filtering procedures contain all information (lines, directions to beacons) that is available on line. If accurate off-line maps are available, the robot can (should) use this information to recalibrate its position and orientation in the room. This calibration is done by *matching* the sensed geometric features with lines in the a priori map. Such matching is commonly implemented as a least squares procedure (hence, Kalman filters could be used to do the job). Figure 12.6 shows a really executed motion of LiAS through the PMA lab, avoiding obstacles, recalibrating its absolute pose by means of sensed beacons or matched line segments.

Reactive planner for obstacle avoidance

The previous paragraphs dealt with how to process the raw sensor signals in order to be able to make a map of the environment, and to adapt it to any changes that occur. However, the frequency with which the maps can be updated and an adapted motion can be calculated is still too low to react to sudden changes in front of the moving vehicle. Hence, some sensor processing has to be present to make the robot react safely to these situations. LiAS has been given a *rule-based fuzzy controller* to do the job: the raw sensor data are given to the fuzzy controller, who reacts to them with a high bandwidth, using a set of user-specified rules (e.g., "Reduce speed if obstacle comes nearer"). Figure 12.7 shows the controller in action, while bringing the robot (autonomously!) out of a dead-end trap.

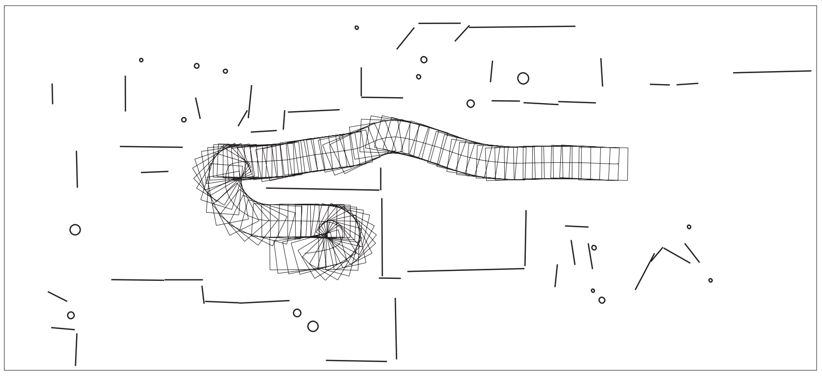


Figure 12.7: The reactive obstacle avoidance fuzzy controller makes the mobile robot escape from a trap obstacle. (Figure courtesy of J. Vandorpe.)

12.4 Map navigation

The previous Section used the Kalman Filter to estimate the state of a system, and to track its evolution. The basic assumptions are that (i) the state doesn't change too fast, and the system is a Markov process (i.e., the current state contains all information collected in the past). This Section treats the case that states do change drastically, but within a given set of possible alternatives; the following Section treats the most complicated case, i.e., a Markov process model is a bad model for the system under observation.

This Section is titled "map navigation," although the problem is more general than that of a mobile robot that has to find its way in a given map, as well as to estimate its current position within this map, e.g., [7]. Other examples are:

- *Monitoring a force-controlled assembly task*, Figs 12.8 and 12.9, [16]. The different possible contact situations are *known*, but the controller doesn't know in advance which sequence of contacts it will encounter, and what the exact contact geometry will be. However, the task has been executed several times already. From this "learning phase," two important sets of probability distributions have been derived: (i) the transition probabilities from one state to a set of other states; and (ii) the typical measurements corresponding to each contact situation.

- *Distance sensing based object recognition and matching*, [9]. A robot moves an ultrasound sensor over an object, and detects where its boundaries are. These measurements are compared to a set of given geometrical models. Each of the models can be given some a priori "weight."

The basic Bayesian algorithm used for this kind of applications is the *Baum-Welch* algorithm for *Hidden Markov Models*, [4, 5, 20, 23]. It is a mixed discrete-continuous model, where the discrete "hidden" variable represents which state/model the system is in, and the continuous variables model the measurements in that state. The estimations are then performed by maximizing the likelihoods of the data given the models.

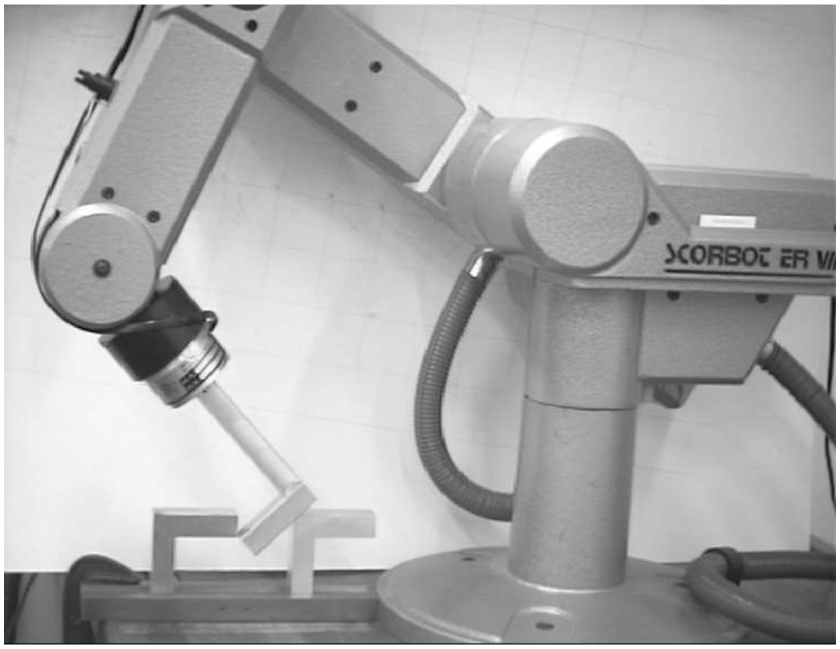


Figure 12.8: Force-controlled assembly task. (Figure courtesy of B. McCarragher.)

12.5 Map building

Kalman Filtering is the simplest level of "intelligent" control: the "map" is given (it consists of only the one object that the filter has to track), and the estimation can be done recursively. The Hidden Markov Model algorithm is one level more complicated: the controller has to find its current position in the map itself. But the job can become even more difficult when the robot has to make its map itself. A nice example of map building by a mobile robot is described in [30, 31, 32]. These references use the *EM algorithm*, [2, 11, 20, 27]. This is an iterative, two-step algorithm:

Expectation step. Given the last estimate of the map, what are the (maximum likelihood) estimates of the position of the robot during all its previous measurements. The position estimates are functions of the statistical parameters that describe the last estimate of the given map.

Maximization step. Given the last estimates of the robot positions, found in the E-step, what are the (maximum likelihood) estimates of the statistical parameters describing the map.

This iteration is executed until convergence. The "position measurements" used in the EM algorithm correspond to "landmarks." The map is represented by assigning probabilities to each cell in a discretized grid of the environment; the probability of a cell reflects the number of times a landmark has been observed in that cell.

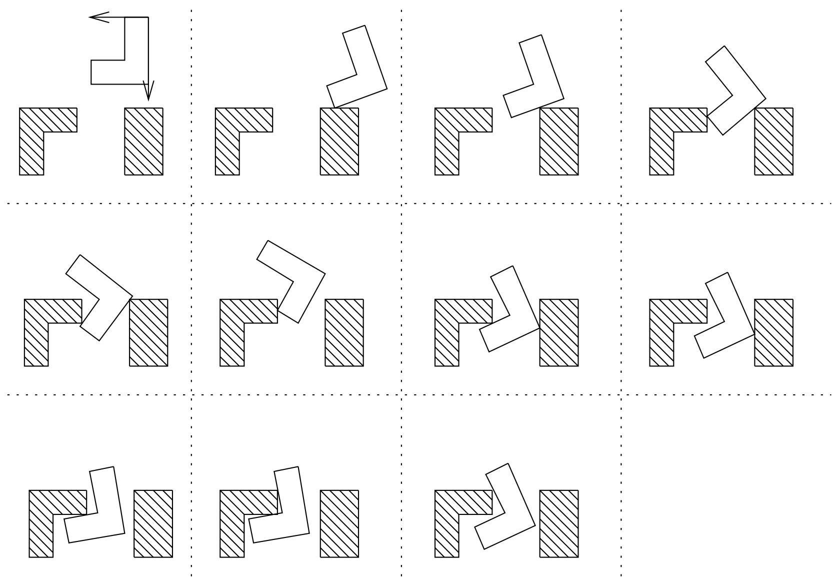


Figure 12.9: Different contact situations in the force-controlled assembly task of Fig. 12.8. (Figure courtesy of B. McCarragher.)

We don't give the details of the Bayesian probability calculus that lies behind this EM algorithm. The calculations are not really very advanced (i.e., they rely on nothing more complicated than Bayes' rule), but they are quite lenghty. The interested is referred to the cited references. However, note that using an exponential family of probability distributions makes the M-step much simpler than in the case of general distributions. Note also that this map building task *cannot* be described by a Markov process: the position estimates in the past will be updated whenever the map is updated! Figures 12.11-fig-sensproc-Thrun2 show the estimated map, and the estimated position of the robot in the map, at two steps in the EM algorithm. In the first Figure, the map has not yet been estimated correctly; in the second Figure, the estimates have converged.

References for this Chapter

[1] S.-I. Amari. *Differential-geometrical methods in statistics*, volume 28 of *Lecture notes in statistics*. Springer Verlag, Berlin, Germany, 1990.

[2] S.-I. Amari. Information geometry of the EM and em algorithms for neural networks. *Neural Networks*, 8(9):1379–1408, 1995.

[3] Y. Bar-Shalom and X.-R. Li. *Estimation and Tracking, Principles, Techniques, and Software*. Artech House, 1993.

[4] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite Markov chains. *Annals of Mathematical Statistics*, 37:1554–1563, 1966.

[5] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41:164–171, 1970.

[6] J. P. Bentley. *Principles of measurement systems*. Longman Scientific & Technical, Harlow, England, 3rd edition, 1995.

[7] W. Burgard, D. Fox, D. Henning, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *14th Nat. Conf. on Artificial Intelligence, AAAI-96*, 1996.

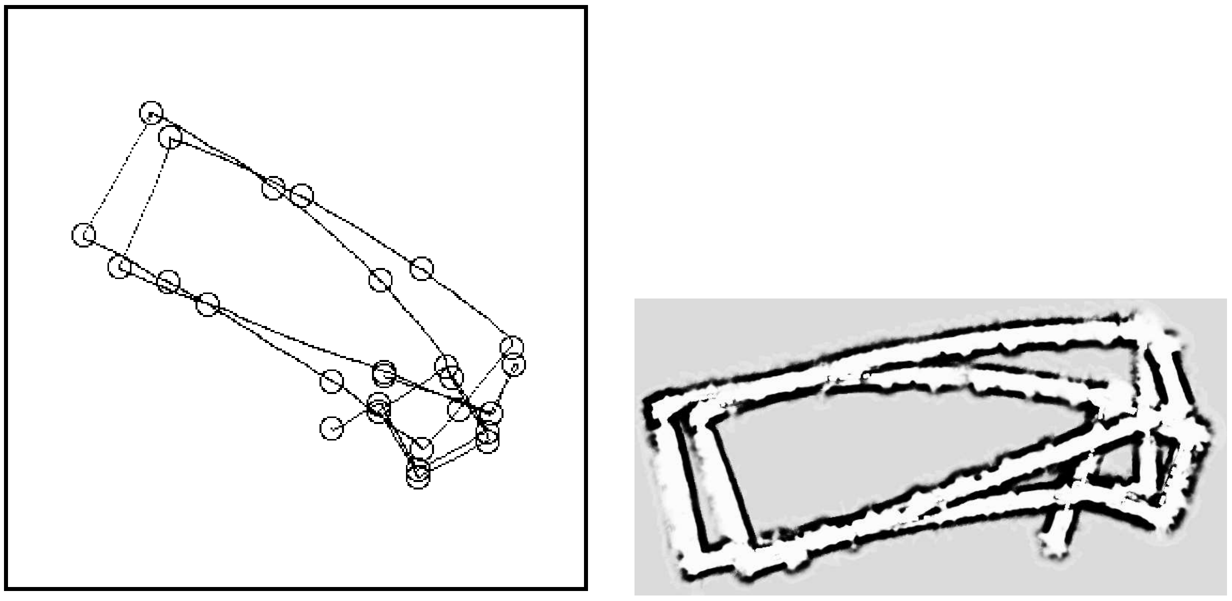


Figure 12.10: Intermediate topological map and occupancy grid.

[8] H. Cox. On the estimation of state variables and parameters for noisy dynamic systems. *IEEE Trans. Autom. Control*, 9(1):5–12, 1964.

[9] J. De Geeter, H. Van Brussel, J. De Schutter, and M. Decréton. Local world modelling for teleoperation in a nuclear environment using a Bayesian multiple hypothesis tree. In *Int. Conf. Intel. Robots and Systems*, pages 1658–1663, Grenoble, France, 1997.

[10] J. De Schutter. Improved force control laws for advanced tracking applications. In *IEEE Int. Conf. Robotics and Automation*, pages 1497–1502, Philadelphia, PA, 1988.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[12] H. F. Durrant-Whyte. Consistent integration and propagation of disparate sensor observations. *Int. J. Robotics Research*, 6(3):3–24, 1987.

[13] G. F. Franklin, J. D. Powell, and M. L. Workman. *Digital Control of Dynamic Systems*. Addison-Wesley, Reading, MA, 2nd edition, 1990.

[14] A. E. Gelb. *Optimal Estimation*. MIT Press, Cambridge, MA, 3rd edition, 1978.

[15] P. J. Hacksel and S. E. Salcudean. Estimation of environment forces and rigid-body velocities using observers. In *IEEE Int. Conf. Robotics and Automation*, pages 931–936, San Diego, CA, 1994.

[16] G. Hovland and B. J. McCarragher. Hidden Markov models as a process monitor in robotic assembly. *Int. J. Robotics Research*, 17:??, 1998.

[17] T. Kailath. An innovations approach to least-squares estimation. Part I: Linear filtering in additive white noise. *IEEE Trans. Autom. Control*, 13(6):646–655, 1968.

[18] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME J. Basic Eng.*, 82:34–45, 1960.

[19] R. E. Kalman and R. S. Bucy. New results in linear filtering and prediction theory. *Trans. ASME J. Basic Eng.*, 83:95–108, 1961.

[20] G. J. McLachlan and T. Krishnan. *The EM algorithm and extensions*. Wiley, New York, NY, 1997.

[21] R. J. Meinhold and N. D. Singpurwalla. Understanding the Kalman-Filter. *The American Statistician*, 37:123–127, 1983.

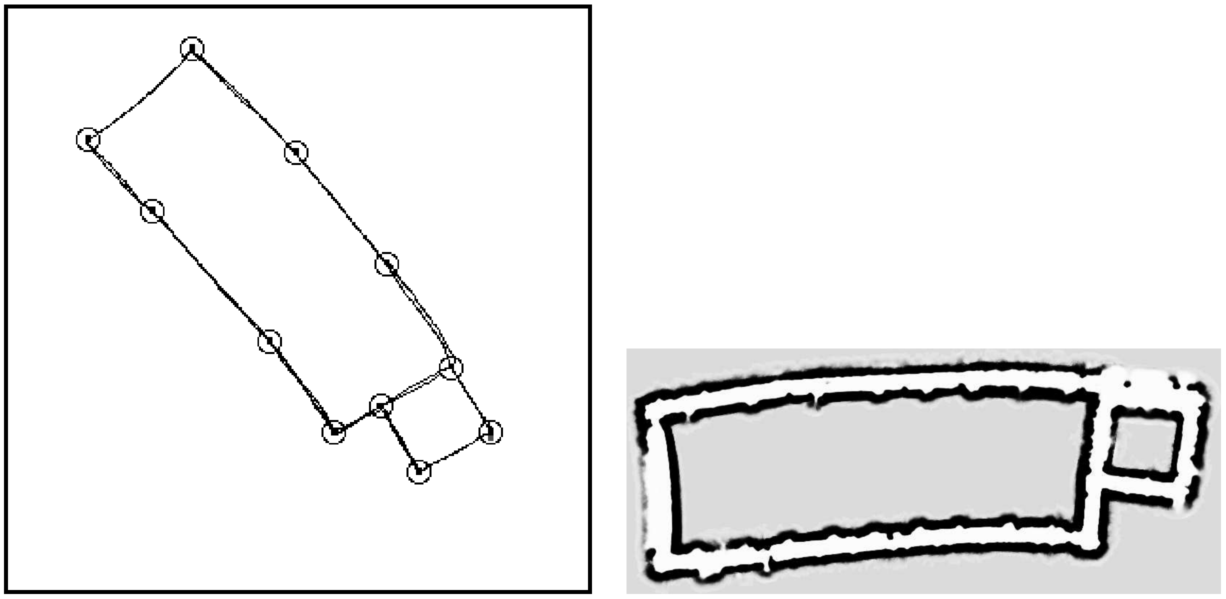


Figure 12.11: Final topological map and occupancy grid.

[22] S. Nicosia and P. Tomei. Robot control by using only joint position measurements. *IEEE Trans. Autom. Control*, 35(9):1058–1061, 1990.

[23] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[24] J. C. Radix. *Filtrage et Lissage Statistique Optimaux Linéaires*. Cépadues Éditions, Toulouse, France, 1984.

[25] I. J. Rudas and F. L. N-Nagy. Advanced industrial robot control using Extended Kalman Filter. In *Symposium on Robot Control*, pages 73–78, Wien, Austria, 1991.

[26] S. Salcudean. A globally convergent angular velocity observer for rigid body motion. *IEEE Trans. Autom. Control*, 36(12):1493–1497, 1991.

[27] J. L. Schafer. *Analysis of incomplete multivariate data*. Chapman and Hall, London, England, 1997.

[28] H. W. Sorenson. Least-squares estimation from Gauss to Kalman. *IEEE Spectrum*, 7:63–68, 1970.

[29] H. W. Sorenson. *Kalman filtering: theory and application*. IEEE Press, New York, NY, 1985.

[30] S. Thrun, D. Fox, and W. Burgard. A probabilistic approach to concurrent mapping and localization for mobile robots. *Autonomous Robots*, 5(3/4):253–271, 1998.

[31] S. Thrun, D. Fox, and W. Burgard. Probabilistic mapping of an environment by a mobile robot. In *IEEE Int. Conf. Robotics and Automation*, pages 1546–1551, Leuven, Belgium, 1998.

[32] S. Thrun, S. Gutmann, D. Fox, W. Burgard, and B. Kuipers. Integrating topological and metric maps for mobile robot navigation: A statistical approach. In *15th Nat. Conf. on Artificial Intelligence, AAAI-98*, 1998.

[33] C. S. Weaver. Estimating the output of a linear discrete system with Gaussian inputs. *IEEE Trans. Autom. Control*, 8:372–374, 1963.

Chapter 13

Motion and force control

The focus in this Chapter is on motion and force control of *serial* chains of rigid bodies, since (i) parallel robots can be approximated as one single moving rigid body (the "legs" are relatively light, and the motors are fixed to the base), and (ii) mobile robots must seldom move in contact with the environment, or at speeds where their dynamics become dominant. Even then the most important dynamic effects (deformations of the tires and the suspensions) are very nonlinear and hence not faithfully described by the dynamic model of a kinematic chain of rigid bodies.

The Chapter consists of two main parts: the first Sections present the most common *free space* controllers (i.e., when the robot is not in contact with the environment); the following Sections explain how to extend these controllers in order to deal with contacts. This Chapter looks at the *servo control* level only, i.e., the feedback loops are closed around the sensor signals without any higher-level "intelligent" sensor processing techniques. Of course, both control levels are complementary; the servo routines typically run at a much higher frequency than the high-level feedback loops.

Section 13.1 repeats some basic properties of first and second order dynamical systems: time constant, natural frequency, damping. These suffice to understand all control algorithms discussed in this Chapter.

Sections 13.2–13.4 describe the most common *free space* motion controllers. Their purpose is to generate actuator commands that make the robot follow the desired position, velocity, and/or acceleration as accurately as possible. These desired motions can be defined in Cartesian space, or in joint space; the transformations between both spaces are given by the kinematic routines presented in earlier chapters.

If the robot is in contact with the environment, the contact reduces the robot's motion degrees of freedom and generates reaction forces. Hence, the controller must be extended with a force control loop. Force control is classified into two categories: passive force control (Sect. 13.5.1) and active force control; active force control, in turn, has two major paradigms: hybrid control (Sect. 13.6) and impedance control (Sect. 13.7).

Basic ideas of this Chapter Controlling the motion of a robot *in free space* is done by either analog (and hence high-bandwidth) *velocity feedback* loops, or by a digital controller using the robot's dynamic equations as *acceleration feedforward* and/or *linearizing* feedback. The former are decoupled (decentralized, independent) and hence much simpler and faster than the centralized approach based on a dynamic model.
A robot that is *constrained* in its motion, is controlled in either the *hybrid* control paradigm, or the *impedance* control paradigm.

13.1 Control theory basics

This Section gives the basic concepts of the theory of *linear* dynamical systems, i.e., systems whose evolution in time is described by *ordinary differential equations* (ODE) of the following form:

$$a_n \frac{d^n y(t)}{dt^n} + \cdots + a_1 y(t) + a_0 = b_m \frac{d^m x(t)}{dt^m} + \cdots + b_1 x(t) + b_0. \tag{13.1}$$

$x(t)$ is the system *input* (or *independent variable*), and $y(t)$ is the system *output* (or *dependent* variable).

13.1.1 Second-order systems

The most complicated ODE that will be used in this Chapter is the dynamic relationship between the joint forces and the end-effector motion:

$$\boldsymbol{M}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{c}(\dot{\boldsymbol{q}}, \boldsymbol{q}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}. \tag{13.2}$$

This equation is of *second order*, with only a zero order "driving" term on the right-hand side. Note that it is a *vector* equation, i.e., it contains a vector of dependent variables $\boldsymbol{q}$. One-dimensional second-order systems are usually written in the following standard form:

$$\ddot{y}(t) + 2\zeta\omega_n \dot{y}(t) + \omega_n^2 y(t) = 0. \tag{13.3}$$

The physical interpretation of ω_n ("*natural frequency*") and ζ ("*damping ratio*") is shown in Fig. 13.1: ω_n determines the number of oscillations per unit of time, and ζ determines how fast the oscillations die out.

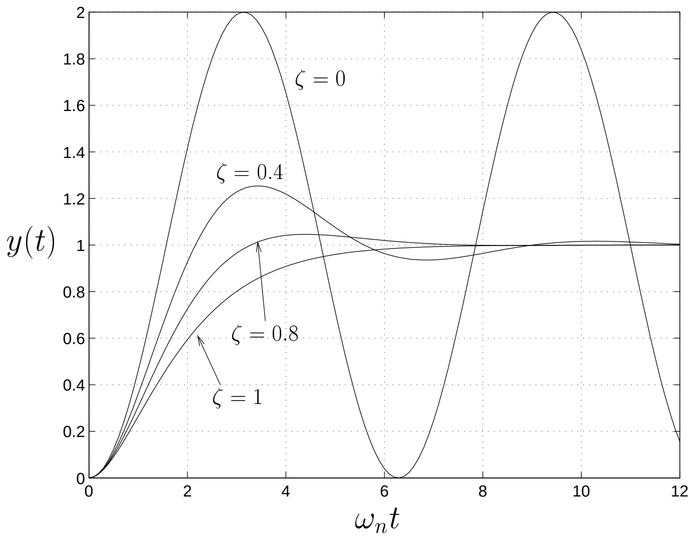


Figure 13.1: Response of a second-order dynamic system to a unit step. A higher *natural "frequency"* ω_n reflects a faster response of the system; the response curves have smaller oscillations if the *damping ratio* ζ is higher. There is no overshoot if $\zeta \geq 1$ ($\zeta = 1$ corresponds to "critical damping").

13.1.2 First-order systems

First-order dynamic systems will also appear in this Chapter. Their standard expression is

$$\dot{y}(t) + \frac{1}{\tau} y(t) = 0. \tag{13.4}$$

Their time evolution is completely determined by the *time constant* τ, Fig. 13.2.

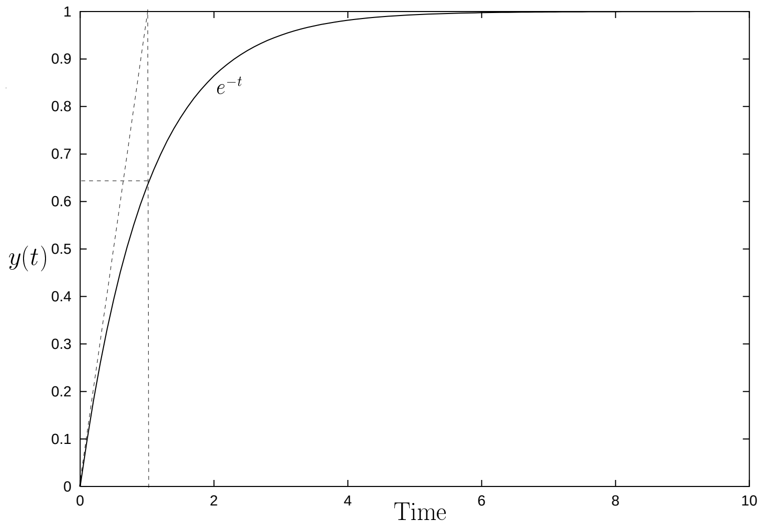


Figure 13.2: Response of a first-order dynamical system to a unit step. The figure shows two ways to deduce the system's *time constant* τ.

13.1.3 Control

Figures 13.1 and 13.2 show the "open loop" time evolution of first and second order systems, i.e., the input to the system (just a step in the above-mentioned Figures) is not changed as a function of the current state of the system. The goal of *control* is to make the input dependent on the state, in order to adapt the "closed-loop" behaviour of the system to the user-defined goals. "Closed-loop" means that the "x" steering input in Eq. (13.1) is calculated as a function of the "y" state, which in turn changes this state, which in turn changes the input, and so on. *Feedback* is the name of the control action that depends on the current *measurements* performed on the system: a control action is generated based on the error between the desired motion and the current measurements of position and velocity. Feedback changes the natural frequency, damping ratio, and/or time constant. *Feedforward* is the name of the control action that uses the *desired motion* of the system, together with its *dynamical model* ("inverse dynamics"), to calculate which forces are required to realize this motion; hence, it allows to *avoid* errors, and also to linearize the system dynamics (Sect. 13.4). Feedforward does not change the natural frequency, damping ratio, and/or time constant. The use of feedback and feedforward will become clear in the following Sections. However, the *design* of controllers is beyond the scope of this text; see, for example, [7, 10, 25].

13.2 One-dimensional motion control

The motion control of a single mass, along one translational degree of freedom, is the simplest control problem, but it illustrates all fundamental ideas that are needed in this Chapter: feedback and feedforward, error dynamics, choice of control parameters, influence of disturbances.

13.2.1 System

Figure 13.3 shows the "robot," as well as the block diagram representing its dynamics, i.e., Newton's law of motion $f = m\ddot{x}$. The system is assumed to be *ideal*: no friction, and the force does not deform the rigid body.

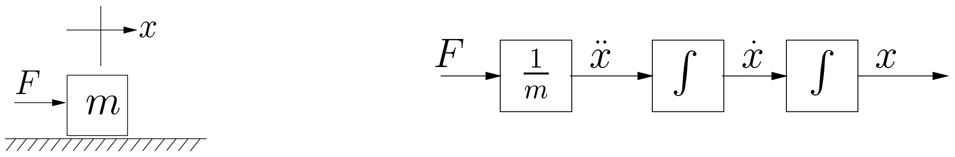


Figure 13.3: One-dimensional "robot." Left: the force F moves the mass m in the x-direction. Right: system model of Newton's law of motion.

13.2.2 Classical ("PID") control

Figure 13.4 shows the classical approach towards controlling the system of Fig. 13.3. The actual and desired positions x and x_d are compared, and their difference (the *position error* e) is multiplied by a (constant, but not physically dimensionless!) "*proportional control gain*" k_p in order to define the force F_{act} that the actuator has to deliver to the moving body.

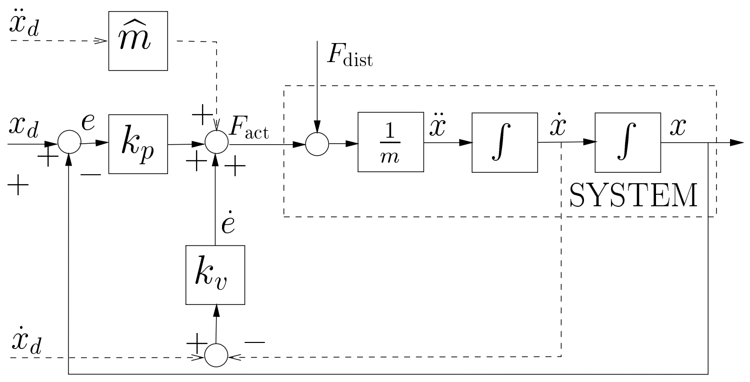


Figure 13.4: One-dimensional motion control. The desired acceleration $\ddot{x}_d$ serves as a *feedforward* input; $\widehat{m}$ is the estimated mass of the controlled body. Many controllers don't use the desired velocity and acceleration inputs.

The control gain is called "proportional" (the "P" part of the term "PID" in the title of this Section) because the result of the control action is proportional to the position error. The "D" stands for "derivative": k_v is the control gain that generates an actuator force (in addition to the one generated by the position control) from the velocity error $\dot{e} = \dot{x}_d - \dot{x}$, i.e., the derivative of the position error. Older robots have a *tachometer* on each joint, to measure the instantaneous velocity $\dot{x}$; modern robots use only position measurements and calculate the velocity by numerical differentiation.

P and D control gains are not sufficient to assure zero errors: any disturbance acting on the system (e.g., due to friction) is counteracted only *after* an error has occurred. A constant disturbance, such as a constant Coulomb friction term, results in a *steady state* error. A steady state error can be avoided by adding an *integral* control term (the "I" part of "PID"): it multiplies the integral of the position error with an "integral gain k_i" and adds the result to the actuator force. However small the error is, adding an actuator force proportional to the integral of that error will eventually be sufficient to make the system move in the direction that reduces the error. Note, however, that adding the integration makes the closed-loop system into a *third-order* dynamical system, for which no intuitive parameters such as natural frequency, damping or time constant exist. The control scheme in Fig. 13.3 does not include an integral feedback term, and also in the sequel integral feedback will be omitted for simplicity.

The "PID" control of the previous paragraphs is purely a *feedback* control: the controller generates actuator forces only *after* an error has occurred. *Avoiding* those errors can only happen through *feedforward* control: the robot uses the "inverse dynamics" model to estimate *in advance* what forces it will require at its motors in order to give the mass of the robot the desired motion. In the one-dimensional case, the inverse dynamics model is simply the estimated mass $\widehat{m}$.

13.2.3 Choosing control gains

The following closed-loop relationship follows immediately from Fig. 13.4 (if no disturbance force appears, and no desired velocity and acceleration inputs are used):

$$m\ddot{x} = k_p(x - x_d) - k_v\dot{x}, \quad \text{or} \quad m\ddot{x} + k_v\dot{x} + k_p x = k_p x_d. \tag{13.5}$$

Comparison with the standard-form equation (13.3) yields the following natural frequency and damping:

$$\omega_n = \sqrt{\frac{k_p}{m}}, \qquad \zeta = \frac{1}{2}\frac{k_v}{\sqrt{k_p m}}. \tag{13.6}$$

The k_p and k_v control gains must be chosen very carefully, keeping in mind the trade-off between *stability* of the system on the one hand, and dynamic closed-loop *performance* on the other hand. k_p should be as large as possible, in order to increase the bandwidth ("natural frequency") of the closed-loop control system; however, ω_n should be significantly smaller than the *mechanical* natural frequency of the robot, or else it will make the system vibrate. k_v should then be chosen such as to keep ζ between 0.7 and 1.0 ("critically damped"), Fig. 13.1. The physical interpretation of the control gains k_p and k_v also follows from Eq. (13.5):

- In steady state, i.e., when all transients have died out and hence all time derivatives are zero, Eq. (13.5) reduces to $k_p(x_d - x) = F_{\text{dist}}$, where F_{dist} is a static *disturbance force*. Hence, k_p has the dimensions of a *stiffness*, and hence it is often called *servo stiffness*.

- Similarly, if $k_p = 0$, then in steady state Eq. (13.5) reduces to $k_v\dot{x} = F_{\text{dist}}$. k_v is seen to have the dimensions of a mechanical damping; hence, it is often called *servo damping*.

13.2.4 Error dynamics

Figure 13.4 shows that the total control signal is:

$$F_{\text{act}} = \widehat{m}\ddot{x}_d + k_v(\dot{x}_d - \dot{x}) + k_p(x_d - x). \tag{13.7}$$

This F_{act} acts on the system with mass m. Assuming that $\widehat{m} = m$, the following linear second-order differential equation in the error $e = x_d - x$ results:

$$m\ddot{e} + k_v\dot{e} + k_p e = 0. \tag{13.8}$$

This means that (i) the error has the same dynamics as the system in Eq. (13.5), and (ii) the control gains k_p and k_v depend on the mass m for a given choice of bandwidth ω_n and damping ratio ζ. If the actuator force doesn't completely cancel the dynamics of the controlled system because a *disturbance force* F_{dist} acts on the system, the *steady state error* is

$$e_{ss} = -\frac{F_{\text{dist}}}{k_p}. \tag{13.9}$$

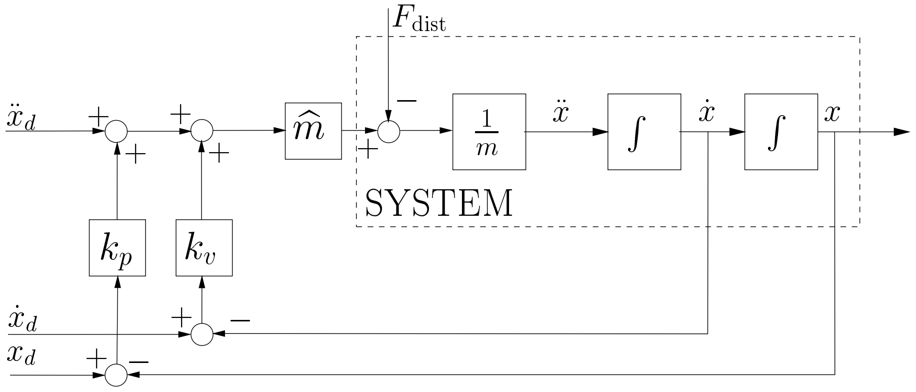


Figure 13.5: One-dimensional motion control with the "dynamic model" in the control loop.

13.2.5 Dynamic model in the loop

In order to make the control constants k_p and k_v independent of the body's mass, the "dynamic model" (i.e., the estimated mass $\widehat{m}$) has to be brought *into* the control loop, Fig. 13.5. The control gains and error dynamics are found in completely the same way as in the previous Section. This yields a mass-independent result:

$$\ddot{e} + k_v\dot{e} + k_p e = 0, \tag{13.10}$$

$$\omega_n = \sqrt{k_p} \rightarrow k_p = \omega_n^2, \tag{13.11}$$

$$\zeta = \frac{1}{2}\frac{k_v}{\sqrt{k_p}} \rightarrow k_v = 2\zeta\sqrt{k_p}. \tag{13.12}$$

The *steady state error* [cf. Eq. (13.9)] is

$$e_{ss} = -\frac{F_{\text{dist}}}{m\,k_p}. \tag{13.13}$$

13.3 Velocity resolved control

The previous Section dealt with the one-dimensional motion of one single rigid body. Robots, however, consist of multiple rigid bodies. The simplest approach to this multi-body motion control problem is to consider each joint as one rigid body, *independent* of the other joints. That is, one neglects the *coupling* effects of the motion of the other links and joints. Figures 13.6 and 13.7 show the resulting controllers, in the single and multiple degrees of freedom cases, respectively.

The inner *velocity* feedback loop is usually implemented in analog hardware, whose high bandwidth reduces the coupling effects between different joints. In the ideal case, the inner loop controls the physical system in such a way that the measured output $\dot{\boldsymbol{q}}$ of the inner loop equals the desired input $\boldsymbol{V}_q$ coming from the outer loop. That is, the dynamics of the inner loops can be neglected. The outer *position* feedback loops are digital controllers. Because the outer loop steers the inner feedback loop with velocity signals, this control scheme is often called *velocity resolved* or *rate resolved* control, [22, 39, 40, 41]. Another name is *decentralized control*, because joints are treated independently.

The error equation is again straightforward from the control schemes. For example, the dynamic behaviour of the error $e_i = q_{i,d} - q_i$ of the control scheme in Fig. 13.6 is given by the following first-order dynamic system:

$$\dot{e}_i + k_{p_i} e_i = 0. \tag{13.14}$$

Its time constant is $\tau = 1/k_{p_i}$. Hence, the dynamics fo the closed loop system are completely governed by the position feedback constant.

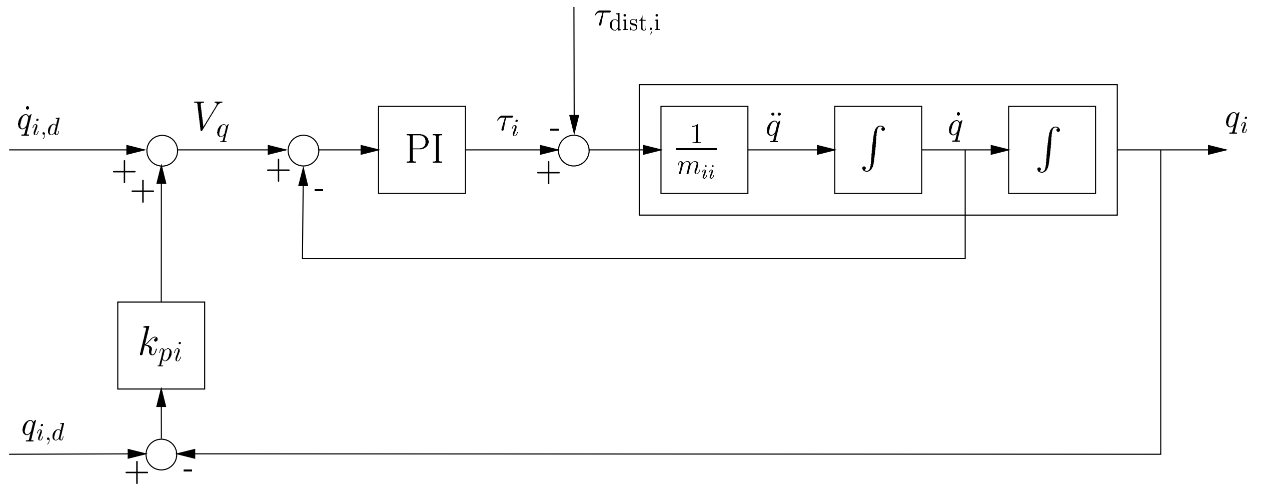


Figure 13.6: Independent joint position control (single degree of freedom case).

13.4 Acceleration resolved control

This method (Fig. 13.8) relies on the knowledge of the complete dynamic model for the robot, as given in Eq. (13.2). Figure 13.8 (Cartesian space) and 13.9 (joint space) show that the errors on position and velocity level are transformed into a feedback at the joint *acceleration* level. Therefore, these methods (for free space as well as constrained motion) often go under the name of *acceleration resolved* methods, [20, 23]. The major features of this control scheme are:

- The coupling between the different links is explicitly taken into account by one single module, i.e., the inverse dynamics ("ID") of the robot. (Therefore, acceleration resolved control is often also called *computed torque*

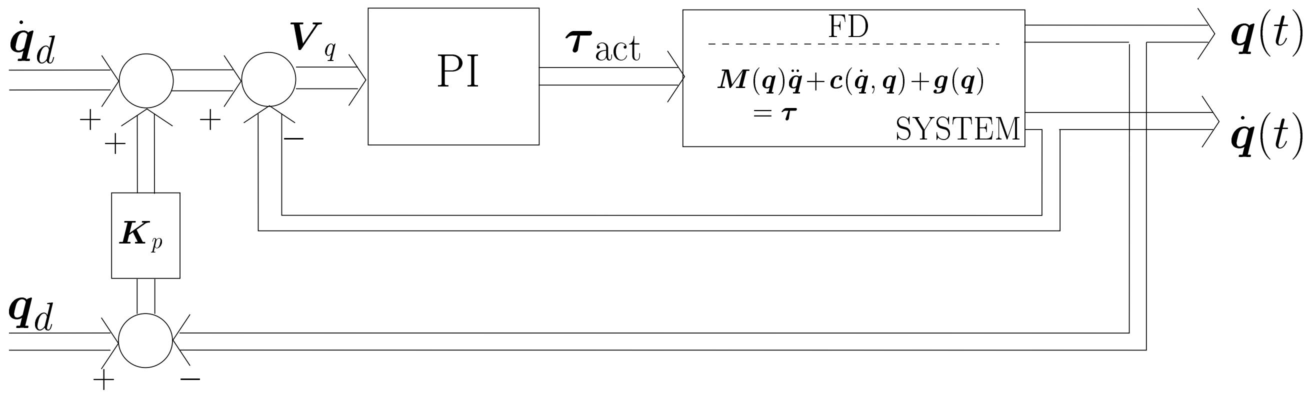

$\dot{\boldsymbol{q}}_d$
$\boldsymbol{V}_q$
PI
$\boldsymbol{\tau}_{\text{act}}$
FD
$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}+\boldsymbol{c}(\dot{\boldsymbol{q}},\boldsymbol{q})+\boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}$
SYSTEM
$\boldsymbol{q}(t)$
$\dot{\boldsymbol{q}}(t)$
$\boldsymbol{K}_p$
$\boldsymbol{q}_d$

Figure 13.7: Independent joint position control (multi-degree of freedom case).

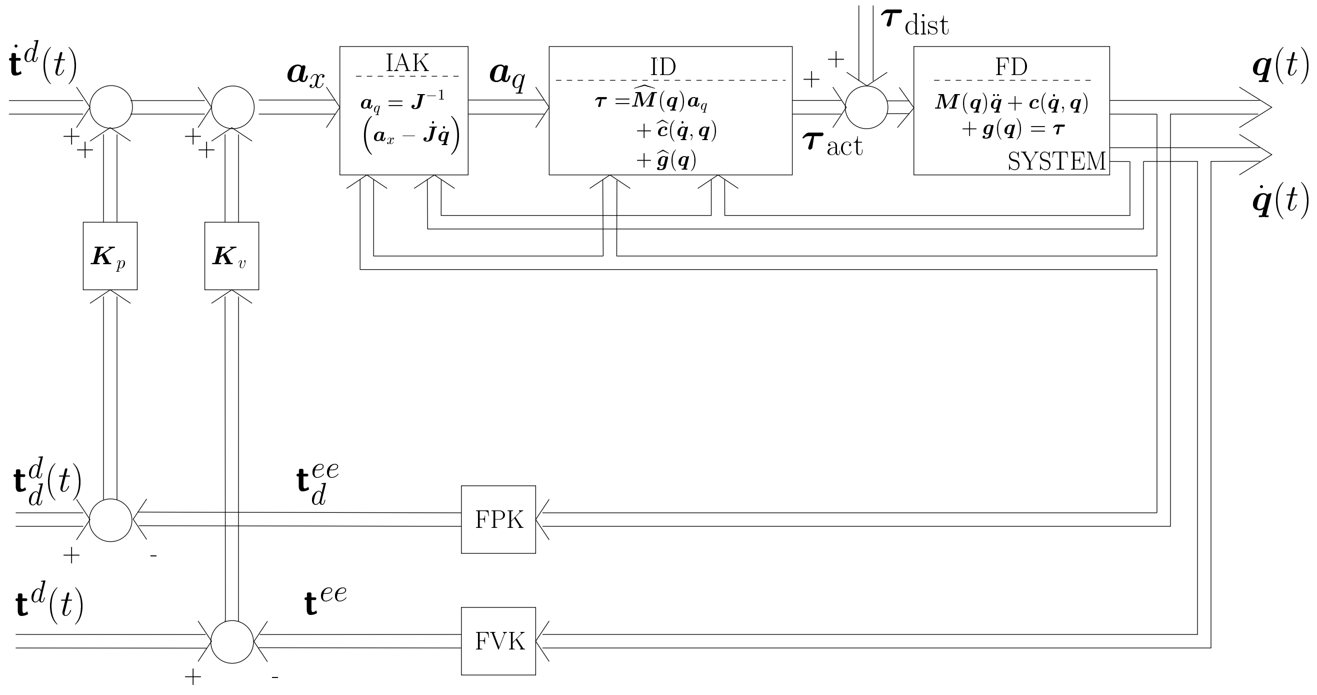
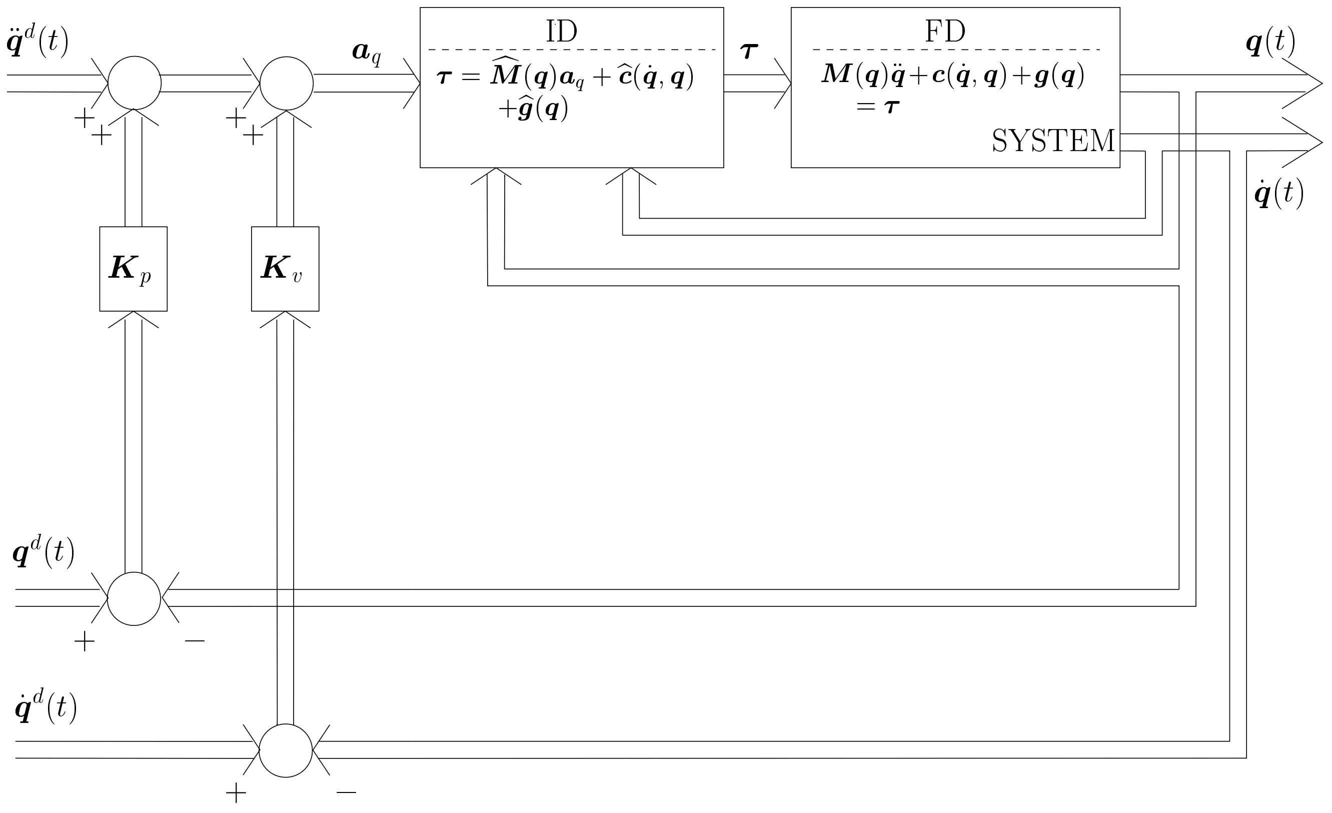

$\dot{\mathbf{t}}^d(t)$
$\boldsymbol{a}_x$
IAK
$\boldsymbol{a}_q = \boldsymbol{J}^{-1}\left(\boldsymbol{a}_x - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right)$
$\boldsymbol{a}_q$
ID
$\boldsymbol{\tau} = \widehat{\boldsymbol{M}}(\boldsymbol{q})\boldsymbol{a}_q + \widehat{\boldsymbol{c}}(\dot{\boldsymbol{q}},\boldsymbol{q}) + \widehat{\boldsymbol{g}}(\boldsymbol{q})$
$\boldsymbol{\tau}_{\text{dist}}$
$\boldsymbol{\tau}_{\text{act}}$
FD
$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\dot{\boldsymbol{q}},\boldsymbol{q}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}$
SYSTEM
$\boldsymbol{q}(t)$
$\dot{\boldsymbol{q}}(t)$
$\boldsymbol{K}_p$
$\boldsymbol{K}_v$
$\mathbf{t}^d_d(t)$
$\mathbf{t}^{ee}_d$
FPK
$\mathbf{t}^d(t)$
$\mathbf{t}^{ee}$
FVK

Figure 13.8: Cartesian-space motion control scheme with full dynamic model in the control loop.

control, or *centralized control*.) The ID module uses the *estimated* parameters $\widehat{\boldsymbol{M}}$, $\widehat{\boldsymbol{c}}$ and $\widehat{\boldsymbol{g}}$. If these parameters are correct, the acceleration resolved approach gives very small tracking errors. Anyhow, it requires a high-quality *identification* of the dynamic parameters (e.g., [36]), and a powerful control computer. Extensions to the presented control scheme adapt the estimates of the dynamic parameters on line, as well as the gain matrices. These advanced controllers are called *adaptive controllers*, [3, 34].

- The inverse acceleration kinematics ("IAK") are needed, to transform the desired acceleration $\boldsymbol{a}_x$ (*after* control)



$\ddot{\boldsymbol{q}}^d(t)$
$\boldsymbol{a}_q$
ID
$\boldsymbol{\tau} = \widehat{\boldsymbol{M}}(\boldsymbol{q})\boldsymbol{a}_q + \widehat{\boldsymbol{c}}(\dot{\boldsymbol{q}}, \boldsymbol{q}) + \widehat{\boldsymbol{g}}(\boldsymbol{q})$
$\boldsymbol{\tau}$
FD
$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\dot{\boldsymbol{q}}, \boldsymbol{q}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}$
SYSTEM
$\boldsymbol{q}(t)$
$\dot{\boldsymbol{q}}(t)$
$\boldsymbol{K}_p$
$\boldsymbol{K}_v$
$\boldsymbol{q}^d(t)$
$\dot{\boldsymbol{q}}^d(t)$
+
−

Figure 13.9: Joint-space motion control scheme with full dynamic model in the control loop.

into the corresponding accelerations of the joints, $\boldsymbol{a}_q$.

- The result of a good working "*inner loop*" (between $\boldsymbol{a}_x$ on the one hand, and the joint positions $\boldsymbol{q}$ and joint velocities $\dot{\boldsymbol{q}}$ on the other hand) *linearizes* the "system": the actual acceleration $\dot{\mathbf{t}}^{ee}$ equals the acceleration $\boldsymbol{a}_x$ desired by the output of the "*outer loop*," i.e.,

$$\dot{\mathbf{t}}^{ee} = \boldsymbol{a}_x, \tag{13.15}$$

  which is a *linear* ODE. This approach of turning a nonlinear physical system into a linear closed-loop system is called *feedback linearization*, [15, 24].

  The outer loop must now cope with an (almost) linear system only. It uses simple position feedback: the position error vector $\boldsymbol{e} = \mathbf{t}_{d,d} - \mathbf{t}_d^{ee}$ is multiplied by the control gain matrix $\boldsymbol{K}_d$, and results in a Cartesian acceleration. (The first subscript "d" in $\mathbf{t}_{d,d}$ and $\mathbf{t}_d^{ee}$ indicates that the twists are to be interpreted as *finite displacement* twists; the second "d" stands for "desired.") The controllers also use velocity feedback (via $\boldsymbol{K}_v$), and sometimes they use acceleration feedforward.

- The computational load of using a full dynamic model in the inner control loop is high. But, since parameters such as the Jacobian matrix and the mass matrix do not change much in the small period (typically 10^{-4} second) between two control instants, some controllers take the *nominal* ("desired") joint values as inputs for the dynamic model and/or re-calculate the inverse dynamics less frequently.

Fact-to-Remember 67 (Summary of motion control schemes)

	decentralized	*centralized (joint space)*	*centralized (Cartesian space)*
computational complexity	$+$	$-$	$--$
high speed performance (dynamics!)	$0/-$	$+$	$+$
low speed performance (friction!)	$+$	$0/-$	$0/-$

- The control gain matrices are usually chosen *diagonal.* This means that all directions (i.e., the different components of either the Cartesian end-effector twist or the joint velocity vector) are *decoupled.* Of course, this decoupling holds only in the ideal case. Non-ideal effects (e.g., disturbance forces, such as friction; inaccurate dynamic or kinematic parameters) cause coupling between the different directions.

- The position error is now a *vector* equation, which one finds in the same way as in the one-dimensional case:

$$\ddot{\boldsymbol{e}} + \boldsymbol{K}_v \dot{\boldsymbol{e}} + \boldsymbol{K}_p \boldsymbol{e} = -\boldsymbol{M}^{-1} \boldsymbol{\tau}_{\text{dist}}. \tag{13.16}$$

  Hence, the *steady state error* [cf. Eq. (13.9)] becomes:

$$\boldsymbol{e}_{ss} = -(\boldsymbol{M}\boldsymbol{K}_p)^{-1} \boldsymbol{\tau}_{\text{dist}}. \tag{13.17}$$

13.5 Active and passive force feedback

Robots of the first generations were conceived as *positioning devices* that operate in "*open loop*," i.e., with little or no feedback at all from the process in which they participate. For high-accuracy assembly tasks, this implies that (i) the motion control of the robot must be accurate, and (ii) all parts or subassemblies have to be prepositioned ("fixtured") with a high accuracy. Accurate fixturing requires expensive and rather inflexible peripheral equipment. Providing robots with sensing capabilities can reduce these accuracy requirements considerably. In particular, for industrial assembly, *force feedback* is extremely useful (Sections 13.6 and 13.7): the measured force gives the robot controller information about how the environment constrains the end-effector's motion, such that it can adapt the current motion set-points. But also for other tasks, in which a tool held by the robot has to make controlled contact with a workpiece (deburring, polishing, cleaning, or sensor-based modelling of the environment), it is not a good idea to fully rely on the positioning accuracy of the robot, and force feedback or force control becomes mandatory. Many non-contact tasks also require sensor feedback, in order to avoid undesired collisions between end-effector and environment and/or to keep the end-effector in a specified relative position and orientation with respect to the environment. These *distance control* tasks can be framed into the above-mentioned force control framework: distances (and their time derivatives) are transformed into virtual forces, by multiplying them by virtual stiffness, damping and/or inertia matrices.

13.5.1 Passive force control

In passive force feedback, the trajectory of the robot end-effector is modified by the interaction forces due to the *inherent* compliance of the robot. The compliance may be due to the structural compliance of links, joints and end-effector, or to the compliance of the position servo. Passive force feedback needs no force sensor, and the preprogrammed trajectory of the end-effector is never changed at execution time: the target assembly position is assumed to be more or less known, such that the compliance will absorb any small misalignments that might (and in practice will) occur during the execution of the task. *Programming* the task consists of

1. Specifying a *nominal position trajectory*, during which the robot will be in contact with the environment without generating excessive forces in the inherent compliance. This is a *software* design process.
2. Adjusting the *impedance* of the compliance (i.e., the mechanical poperties of the compliance: number of degrees of freedom, inertia, stiffness and damping) to the task, [30, 35]. This involves a *hardware* design.

Since the peg-in-hole type of task is so common in assembly, robot tool manufacturers sell compliances for these tasks. These devices are known under the name of *RCC*, or Remote Centre Compliance, [6, 8, 43]: they support the peg in such a way that the tip of the peg is a mechanical *centre of compliance*, i.e., forces through this centre are taken up by translations along the same direction as the force, and similarly for torques. This means that the RCC is a passive implementation of the task frame specification of the peg-in-hole task Some remarks about RCCs:

1. Passive compliances can take up only *small* uncertainties in position and orientation. (No absolute quantification of the word “small” exists, though.)
2. They react instantaneously, i.e., much faster than active repositioning by a computer control algorithm based on measured forces.
3. In general, a mechanically compliant element does *not* possess a so-called *centre of compliance*, [2, 18, 19, 27, 26], i.e., no reference frame exists in which the stiffness or compliance matrices are diagonal.
4. RCCs lack flexibility, since for every robot task a special purpose compliant end-effector has to be designed and mounted.
5. Since no forces are measured, an RCC can neither detect nor cope with error conditions involving excessive contact forces, and it is not a guarantee that high contact forces will never occur.

13.5.2 Active force control

In active force feedback, the interaction forces are measured, fed back to the controller, and used to *modify* on-line the nominally specified trajectory of the robot end-effector. Active force feedback has an answer to all above-mentioned disadvantages of passive force feedback, but it is usually slower, more expensive, and more sophisticated. Apart from a force sensor, it also requires an adapted programming and control system. In addition, it has been shown [5] that, in order to obtain a reasonable task execution speed and disturbance rejection capability, active force feedback has to be used in combination with some degree of passive force feedback: feedback, by definition, always comes *after* a motion or force error has occurred, hence some passive compliance is needed in order to keep the reaction forces below an acceptable threshold.

Joint space controllers, or Cartesian controllers with much less than six degrees of freedom were developed first, e.g., [28, 29, 31, 42]; full Cartesian control schemes followed soon, [4, 12, 17, 33]. "Cartesian" means that *end-effector* motions and forces are specified and controlled, not *joint* motions or torques. Of course, the controller eventually transforms the results of the Cartesian control into joint torques, but the user need not bother about these kinematic transformations. The active force feedback literature mainly works with one of the following two motion constraint *models*:

1. Robot and environment are perfectly *rigid*, and the interaction between both is *geometric*, i.e., the robot can move along the environment without deforming neither the environment nor itself.

2. Robot and environment are perfectly *rigid*, and the interaction between both is "*soft*." Most often, the interaction is modelled by a *mass–spring–damper* system, with the extra simplifying assumptions that (i) the compliance is linear, and (ii) all compliance in the system (including that of the manipulator, its motion control, the force sensor and the tool) is localized in the environment.

These two different motion constraint models have given rise to two different control "paradigms": *hybrid* control (Sect. 13.6) for the "geometric" interaction model, and *impedance* control (Sect. 13.7) for the "soft" interaction model.

13.6 Hybrid control

Hybrid control (or "hybrid force/position control") is the Cartesian control approach that corresponds most closely to the "Compliance Frame" (or "Task Frame") task specification tool, [1, 21], because that also models the motion constraint as completely rigid. The six motion degrees of freedom of the robot end-effector are split into a number $n^v < 6$ of *position* or *velocity-controlled* degrees of freedom, and $n^f = 6 - n^v$ *force-controlled* degrees of freedom. Most papers on force control say that these force- and velocity-controlled directions are "orthogonal." But "orthogonality" is not the correct property, since twists and wrenches belong to physically different spaces, [9]. The appropriate concept is *reciprocity*: an (ideal) contact wrench on the end-effector is reciprocal to any of the possible end-effector twists allowed by the constraint, if moving the end-effector against the wrench requires no power.

13.6.1 System

The robot system to be moved is the same as in the case of free space motion control, but now the actuated force also has to counteract the reaction wrench generated in the robot-environment interaction. Although hybrid control uses the simplification of a rigid, geometric constraint model at the *task specification* level, at the *force control* level it uses a "soft" interaction model. Both systems (robot and interaction) are assumed to be *ideal*: no friction, and the force does not deform the rigid body. This leads to the following extension of the general dynamic equation for an unconstrained robot, Eq. (13.2):

$$\boldsymbol{M}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{c}(\dot{\boldsymbol{q}},\boldsymbol{q}) + \boldsymbol{g}(\boldsymbol{q}) + (\widetilde{\boldsymbol{\Delta}}\,\boldsymbol{J})^T \boldsymbol{w}^{ee} = \boldsymbol{\tau}, \tag{13.18}$$

where the influence of the reaction wrench at the joints is found from the "Jacobian transpose" relationship.The Cartesian wrench $\boldsymbol{w}^{ee}$ is determined by the dynamics of the motion of the robot arm, and by the contact force.

Note that the geometric contact model cannot make distinction between static contact forces, and forces generated by damping or inertia; the control loop itself will take into account the contact impedance, although usually only the compliance part (i.e., the robot-environment interaction is a pure spring).

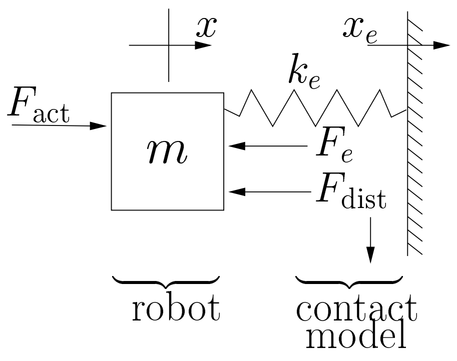


Figure 13.10: Mass–spring model of a one-dimensional force-controlled "robot." The force is controlled in the "left-right" direction; velocity in controlled in the "up-down" direction. The spring is infinitely stiff in the "geometric" constraint model. F_e is the contact force generated by the environment; F_{dist} is any non-modelled extra force (e.g., friction); it can occur in the force-controlled direction, as well as in the velocity-controlled direction.

13.6.2 One-dimensional force control

The motion and force control of a single mass, along one translational degree of freedom, is the simplest possible force control problem. As was the case with the one-dimensional motion control, it illustrates the fundamental idea: a force error in a force-controlled direction is translated into a motion that interacts with the robot-environment mass–spring–damper in such a way as to reduce the force error. In the simplest model, this robot-environment impedance is only a spring; this simplification is often quite accurate, because, almost by definition, the robot moves only slowly when in contact with the environment.

System. The dynamics of the one-dimensional system in Fig. 13.10 are given by

$$m\ddot{x} = F_{\text{act}} - F_{\text{dist}} - F_e, \tag{13.19}$$

under the assumption that the environment doesn't move, i.e., $\dot{x}_e = 0$. F_e is the force generated by the mass–spring–damper system that models the contact interaction.

Control. The one-dimensional force control scheme is given in Fig. 13.11. The "robot" dynamics are modelled by a mass $\widehat{m}$, and the robot-environment interaction by a spring with spring constant $\widehat{k}_e$. The interaction force F_e is found as the deformation of the *real* interaction compliance k_e: $F_e = k_e(x - x_e)$. The dynamics of the force error $e_f = F_d - F_e$ are straightforwardly derived, under the assumption that the modelled dynamics ($\widehat{k}_e$ and $\widehat{m}$) are equal to the real dynamics (k_e and m):

$$\ddot{e}_f + k_{vf}\,\dot{e}_f + k_{pf}\,e_f = 0, \tag{13.20}$$

or, in case of a disturbance

$$\ddot{e}_f + k_{vf}\,\dot{e}_f + k_{pf}\,e_f = \frac{F_{\text{dist}}}{m\,k_e^{-1}}. \tag{13.21}$$

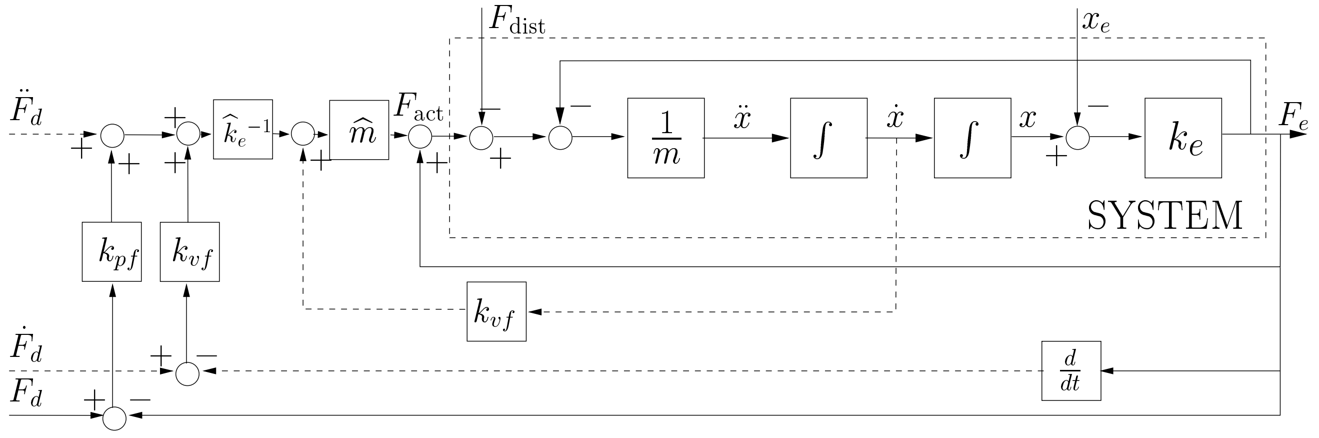


Figure 13.11: One-dimensional force control scheme, when the robot-environment interaction model is a spring with (modelled) stiffness $\widehat{k}_e$.

Again, this is a second-order system, whose control constants are chosen in a way similar to the motion control case. The *steady state* error $e_{f,ss}$ is found from Eq. (13.21) by putting all time derivatives to zero:

$$e_{f,ss} = \frac{F_{\text{dist}}}{m\, k_{pf}\, k_e^{-1}}. \tag{13.22}$$

Hence, disturbance forces are "attenuated" by (i) the controller (k_{pf}), (ii) the system dynamics (m), and (iii) the interaction dynamics (k_e).

The control scheme contains the derivative of the measured force, but because force measurements tend to be quite noisy, this derivative is most often not used. Some implementations replace it by $k_e(\dot{x} - \dot{x}_e)$, but the quality of this "observed" force derivative relies on two assumptions: (i) the environment doesn't move, and (ii) $k_e = \widehat{k}_e$. Also the derivatives of the *desired* forces, $\dot{F}_d$ and $\ddot{F}_d$, are most often not used.

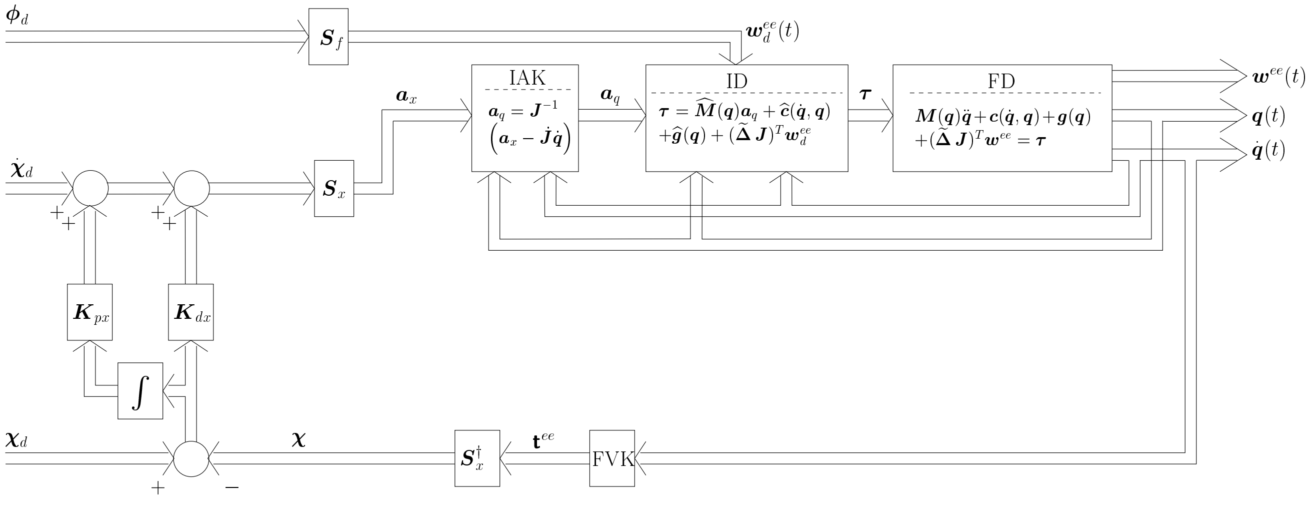
13.6.3 Cartesian force control–Acceleration resolved

This Section discusses full six-dimensional hybrid force/position control, for both rigid (Fig. 13.12) and soft (i.e., purely spring-like, Fig. 13.13) robot-environment interactions. The major differences with the one-dimensional case are:

- As in the motion control case, there is an *inner control loop* that linearizes the system: it makes sure that the *outer loop* can assume that it works with a system whose dynamics are "infinite," i.e., $\dot{\mathbf{t}} = \boldsymbol{a}_x$, with $\dot{\mathbf{t}}$ the *actually executed* end-effector acceleration twist, and $\boldsymbol{a}_x$ the end-effector acceleration twist *desired* by its controller.
- The inverse acceleration kinematics ("IAK") are used to transform all forces and motions between the joint space (where the actual actuation takes place) and the Cartesian space (where the end-effector moves and makes contact with the environment).
- The end-effector's six motion degrees of freedom are divided into a number of velocity-controlled degrees of freedom, and a number of force-controlled degrees of freedom. The sum of both numbers is 6.

- Both velocity-controlled and force-controlled subspaces have a set of basis screws, collected in the matrices $\boldsymbol{S}_x$ and $\boldsymbol{S}_f$, respectively. All *desired* quantities (motions and forces) are specified by giving *coordinates* in these bases: $\boldsymbol{\phi}_d$ for desired forces, and $\boldsymbol{\chi}_d$ for desired velocities. All *measured* quantities (which are six-dimensional!) are *projected* onto their corresponding coordinate subspaces by the pseudo-inverses $\boldsymbol{S}_x^\dagger$ (for the velocity-controlled subspace) and $\boldsymbol{S}_f^\dagger$ (for the force-controlled subspace). Recall that these pseudo-inverses are not uniquely defined.
- The control gain matrices are not six-dimensional, but have dimensions corresponding to the dimension of the subspace on which they act. Note that the control takes place on the *coordinates*, and not in the force- or velocity-controlled subspace itself.

Figures 13.12 and 13.13 show the control schemes, for rigid and soft robot-environment interactions, respectively. Note that the environment is always assumed to be constant, i.e., the bases of the force- and velocity-controlled subspaces do not change over time.



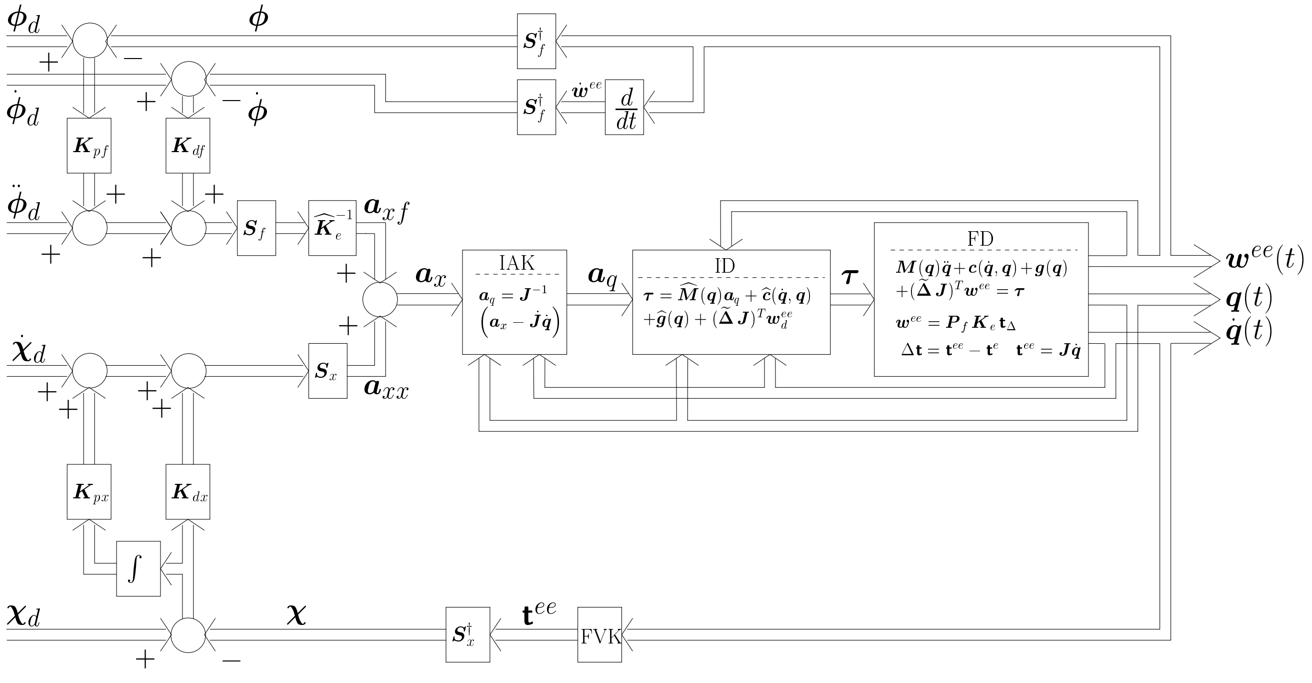
$\boldsymbol{\phi}_d$
$\boldsymbol{S}_f$
$\boldsymbol{w}_d^{ee}(t)$
IAK
$\boldsymbol{a}_q = \boldsymbol{J}^{-1}\left(\boldsymbol{a}_x - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right)$
$\boldsymbol{a}_x$
$\boldsymbol{a}_q$
ID
$\boldsymbol{\tau} = \widehat{\boldsymbol{M}}(\boldsymbol{q})\boldsymbol{a}_q + \widehat{\boldsymbol{c}}(\dot{\boldsymbol{q}},\boldsymbol{q}) + \widehat{\boldsymbol{g}}(\boldsymbol{q}) + (\widetilde{\boldsymbol{\Delta}}\,\boldsymbol{J})^T\boldsymbol{w}_d^{ee}$
$\boldsymbol{\tau}$
FD
$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\dot{\boldsymbol{q}},\boldsymbol{q}) + \boldsymbol{g}(\boldsymbol{q}) + (\widetilde{\boldsymbol{\Delta}}\,\boldsymbol{J})^T\boldsymbol{w}^{ee} = \boldsymbol{\tau}$
$\boldsymbol{w}^{ee}(t)$
$\boldsymbol{q}(t)$
$\dot{\boldsymbol{q}}(t)$
$\dot{\boldsymbol{\chi}}_d$
+
+
+
+
$\boldsymbol{S}_x$
$\boldsymbol{K}_{px}$
$\boldsymbol{K}_{dx}$
$\int$
$\boldsymbol{\chi}_d$
$\boldsymbol{\chi}$
$\boldsymbol{S}_x^\dagger$
$\mathbf{t}^{ee}$
FVK
+
−

Figure 13.12: Cartesian hybrid force/position control scheme, when the robot-environment interaction is modelled as fully rigid.

Rigid environment. A completely rigid robot-environment interaction (Fig. 13.12) is, of course, an idealization. However, it can be a useful approximation in case the forces in the force-controlled subspaces are only "controlled" via feedforward. That is, the *desired* contact wrench $\mathbf{w}_d^{ee}$ is calculated from the specification of the corresponding coordinates $\boldsymbol{\phi}_d$, and introduced as such in the inverse dynamics routine of the inner control loop.

The motion in the velocity-controlled directions control follows the same principles as in the motion control case: the errors are multiplied by control gain matrices. Choosing diagonal gain matrices *decouples* the control between the different base twists or wrenches (under the assumption of small disturbances, accurate models, and a high-bandwidth inner loop). Each decoupled direction will give a second-order system, with classical control design.

Soft environment. In this case, the robot-environment interaction is a *spring* with stiffness matrix $\boldsymbol{K}_e$, which is approximated in the model by a matrix $\widehat{\boldsymbol{K}}_e$. The differences with the rigid interaction case are:



ϕ_d
ϕ
$\dot{\phi}_d$
$\dot{\phi}$
$\ddot{\phi}_d$
$S_f^\dagger$
$\dot{w}^{ee}$
$\frac{d}{dt}$
K_{pf}
K_{df}
S_f
$\widehat{K}_e^{-1}$
a_{xf}
a_x
a_q
τ
IAK
$a_q = J^{-1}(a_x - \dot{J}\dot{q})$
ID
$\tau = \widehat{M}(q)a_q + \widehat{c}(\dot{q}, q) + \widehat{g}(q) + (\widetilde{\Delta}\, J)^T w_d^{ee}$
FD
$M(q)\ddot{q} + c(\dot{q}, q) + g(q) + (\widetilde{\Delta}\, J)^T w^{ee} = \tau$
$w^{ee} = P_f K_e \mathbf{t}_\Delta$
$\Delta\mathbf{t} = \mathbf{t}^{ee} - \mathbf{t}^e \quad \mathbf{t}^{ee} = J\dot{q}$
$w^{ee}(t)$
$q(t)$
$\dot{q}(t)$
$\dot{\chi}_d$
S_x
a_{xx}
K_{px}
K_{dx}
$\int$
χ_d
χ
$S_x^\dagger$
$\mathbf{t}^{ee}$
FVK

Figure 13.13: Cartesian hybrid force/position control scheme, when the robot-environment interaction is modelled as spring-like, with stiffness matrix $\boldsymbol{K}_e$.

1. Force *feedback control* is now possible in the force-controlled subspaces, since forces can be regulated by controlling the deformation of the interaction spring.
2. The "system" is extended with the relationships between (i) the stiffness matrix $\boldsymbol{K}_e$ of the interaction spring, (ii) its deformation $\mathbf{t}_\Delta$ (which is the difference between the end-effector twist $\mathbf{t}^{ee}$ and the environment twist $\mathbf{t}^e$), (iii) the force-controlled subspace (on which the matrix $\boldsymbol{P}_f = \boldsymbol{S}_f \boldsymbol{S}_f^\dagger$ projects the measured end-effector wrench), and (iv) the forward velocity kinematics of the robot (modelled by the Jacobian matrix $\boldsymbol{J}$).

13.6.4 Cartesian force control–Velocity resolved

Again as in the free space motion control case (Sect. 13.3), the inner loop can be implemented as an analog velocity controller, Fig. 13.14. The advantages of the velocity resolved control approach are still valid. And in many respects even more so than in the free space motion case: motion in contact implies slow motion *and* much friction, and this are exactly the two aspects in which the velocity resolved approach excells.

13.7 Impedance control

The *impedance control* approach, [13, 16], differs from the hybrid approach, both in task specification and in control:

1. *Task specification.* Hybrid control specifies desired motion *and* force trajectories; impedance control, specifies (i) a *nominal motion* trajectory (i.e., the desired motion if the environment would be perfectly known), and (ii)

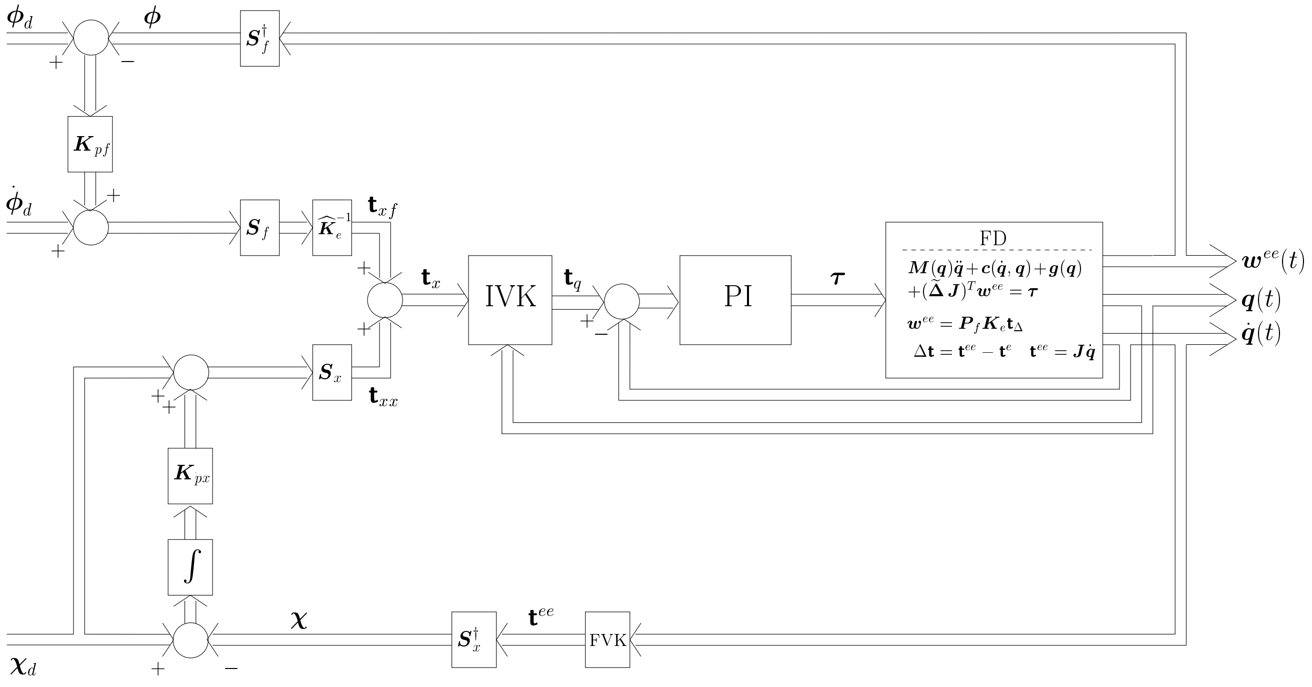


Figure 13.14: Cartesian hybrid force/position control scheme, when the inner loop is implemented as an analog velocity controller.

a desired *dynamic relationship* between, on the one hand, the deviations from this nominal trajectory (caused by the contact with the environment), and, on the other hand, the forces exerted by the environment:

$$\mathbf{w} = -M_d\, \dot{\mathbf{t}} + C_d\, \tilde{\mathbf{t}} + K_d \int \tilde{\mathbf{t}}\, dt. \tag{13.23}$$

$\tilde{\mathbf{t}}$ is the Cartesian error velocity twist, i.e., the difference between the prescribed velocity $\mathbf{t}_d$ and the measured velocity $\mathbf{t}$; $\dot{\mathbf{t}}$ is the Cartesian acceleration twist; M_d, C_d and K_d are *user defined* inertia, damping and stiffness matrices, respectively. (Note that the integral in Eq. (13.23) is in fact only defined if the position errors are always small, and can hence be described by infinitesimal twists.)

Compared to hybrid force/position control the apparent advantage of impedance control is that no explicit knowledge of the constraint kinematics is required. However in order to obtain a satisfactory dynamic behaviour, the inertia, damping and stiffness matrices have to be tuned for each particular task. Hence they embody implicit knowledge of the task geometry, and hence task specification and control are (again) intimately linked.

2. *Control.* For control purposes the dynamic relationship in Eq. (13.23) can be interpreted in two ways. In the first way, it is the model of an *impedance*, i.e., the robot reacts to the "deformations" of its planned position and velocity trajectories by generating forces. Special cases are *stiffness control*, [33], where $M_d = C_d = 0$, and *damping control*, where $M_d = K_d = 0$. Stiffness control is quite popular in *robot hands*: the fingers squeeze the grasped object, and give way to disturbing motions of the objects as if they were pure springs.

   Equation (13.23) can also be interpreted the other way around as an *admittance*, i.e., the robot reacts to the constraint forces by deviating from its planned trajectory *as if* it were a rigid body with the dynamic behaviour prescribed by Eq. (13.23).

Examples of impedance/admittance controllers can be found, among *many* others, in [11, 14, 32, 37, 38].

References for this Chapter

[1] H. Bruyninckx and J. De Schutter. Specification of force-controlled actions in the "Task Frame Formalism": A survey. *IEEE Trans. Rob. Automation*, 12(5):581–589, 1996.

[2] N. Ciblak and H. Lipkin. Centers of stiffness, compliance, and elasticity in the modelling of robotic systems. In *ASME 23rd Biennial Mecahnisms Conference (Vol. DE-72)*, pages 185–195, Minneapolis, MN, 1994.

[3] J. J. Craig, P. Hsu, and S. S. Sastry. Adaptive control of mechanical manipulators. *Int. J. Robotics Research*, 6(2):16–28, 1987.

[4] J. De Schutter. *Compliant Robot Motion: Task Formulation and Control*. PhD thesis, Katholieke Universiteit Leuven, Department of Mechanical Engineering, Leuven, Belgium, 1986.

[5] J. De Schutter and H. Van Brussel. Compliant robot motion II. A control approach based on external control loops. *Int. J. Robotics Research*, 7(4):18–33, 1988.

[6] T. L. DeFazio, D. S. Seltzer, and D. E. Whitney. The instrumented remote center compliance. *The Industrial Robot*, 11(4):238–242, 1984.

[7] R. C. Dorf. *Modern Control Systems*. Addison Wesley, 6th edition, 1992.

[8] S. Drake. Using compliance in lieu of sensory feedback for automatic assembly. Technical Report T-657, Charles Stark Draper Laboratory, 1977.

[9] J. Duffy. The fallacy of modern hybrid control theory that is based on "orthogonal complements" of twist and wrench spaces. *J. Robotic Systems*, 7(2):139–144, 1990.

[10] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Addison-Wesley, Reading, MA, 3rd edition, 1990.

[11] A. Giraud. Generalized active compliance for parts mating with assembly robots. In M. Brady and R. Paul, editors, *Robotics Research, The First International Symposium*, chapter 10, pages 949–960. MIT Press, Cambridge, MA, 1984.

[12] G. Hirzinger. Direct digital control using a force-torque sensor. In *Proc. IFAC Symp. on Real Time Digital Control Applications*, 1983.

[13] N. Hogan. Impedance control: An approach to manipulation. Parts I-III. *Trans. ASME J. Dyn. Systems Meas. Control*, 107:1–24, 1985.

[14] R. L. Hollis, S. E. Salcudean, and A. P. Allan. A six-degree-of-freedom magnetically levitated variable compliance fine-motion wrist: design, modeling, and control. *IEEE Trans. Rob. Automation*, 7(3):320–332, 1991.

[15] A. Isidori. *Nonlinear Control Systems*. Communications and Control Engineering Series. Springer Verlag, Berlin, Germany, 2nd edition, 1989.

[16] H. Kazerooni, P. K. Houpt, and T. B. Sheridan. Robust compliant motion for manipulators, Part II: Design method. *IEEE J. Rob. Automation*, RA-2:93–105, 1986.

[17] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE J. Rob. Automation*, RA-3(1):43–53, 1987.

[18] H. Lipkin and T. Patterson. Geometrical properties of modelled robot elasticity: Part I—decomposition. In *Proceedings of the ASME Conference on Robotics, Spatial Mechanisms, and Mechanical Systems*, pages 179–185, Scottsdale, AZ, 1992.

[19] J. Lončarić. Normal forms of stiffness and compliance matrices. *IEEE J. Rob. Automation*, RA-3(6):567–572, 1987.

[20] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul. Resolved-acceleration control of mechanical manipulators. *IEEE Trans. Autom. Control*, 25(3):468–474, 1980.

[21] M. T. Mason. Compliance and force control for computer controlled manipulators. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-11(6):418–432, 1981.

[22] M. L. Moe. Kinematics and rate control of the Rancho arm. In *Proceedings of the First CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators*, pages 253–272, Wien, Austria, 1973. Springer Verlag.

[23] P. F. Muir and C. P. Neuman. Resolved motion rate and resolved acceleration servo-control of wheeled mobile robots. In *IEEE Int. Conf. Robotics and Automation*, pages 1133–1140, Cincinnati, OH, 1990.

[24] H. Nijmeijer and A. van der Schaft. *Nonlinear Dynamical Control Systems*. Springer Verlag, New York, NY, 1996. Corrected 3rd printing.

[25] K. Ogata. *Modern control engineering*. Prentice-Hall, London, England, 2nd edition, 1990.

[26] T. Patterson and H. Lipkin. A classification of robot compliance. *Trans. ASME J. Mech. Design*, 115:581–584, 1993.

[27] T. Patterson and H. Lipkin. Structure of robot compliance. *Trans. ASME J. Mech. Design*, 115:576–580, 1993.

[28] R. Paul. Modelling, trajectory calculation and servoing of a computer controlled arm. Technical Report AIM-177, Stanford University, Computer Science Department, 1972.

[29] R. P. Paul and B. Shimano. Compliance and force control. In *Proc. Joint Automatic Control Conf.*, pages 694–699, 1976.

[30] M. A. Peshkin. Programmed compliance for error corrective assembly. *IEEE Trans. Rob. Automation*, 6(4):473–482, 1990.

[31] M. Raibert and J. J. Craig. Hybrid position/force control of manipulators. *Trans. ASME J. Dyn. Systems Meas. Control*, 102:126–133, 1981.

[32] S. E. Salcudean, S. Bachmann, and D. Ben-Dov. A six degree-of-freedom wrist with pneumatic suspension. In *IEEE Int. Conf. Robotics and Automation*, pages 2444–2450, San Diego, CA, 1994.

[33] J. K. Salisbury. Active stiffness control of a manipulator in Cartesian coordinates. In *19th IEEE Conf. on Decision and Control*, 1980.

[34] S. S. Sastry and M. Bodson. *Adaptive control: stability, convergence, and robustness*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[35] J. M. Schimmels and M. A. Peshkin. Admittance matrix design for force-guided assembly. *IEEE Trans. Rob. Automation*, 8(2):213–227, 1992.

[36] J. Swevers, C. Ganseman, D. Bilgin, J. De Schutter, and H. Van Brussel. Optimal robot excitation and identification. *IEEE Trans. Rob. Automation*, 13(5):730–740, 1997.

[37] H. Van Brussel and J. Simons. The adaptable compliance concept and its use for automatic assembly by active force feedback accomodations. In *Int. Symp. Industrial Robots*, pages 167–181, Washington, DC, 1979.

[38] H. Van Brussel, H. Thielemans, and J. Simons. Further developments of the active adaptive compliant wrist (AACW) for robot assembly. In *Int. Symp. Industrial Robots*, pages 377–384, Tokyo, Japan, 1981.

[39] K. J. Waldron. Geometrically based manipulator rate control algorithms. *Mechanism and Machine Theory*, 17(6):379–385, 1982.

[40] H. Weiss. Quaternion-based rate/attitude tracking system with application to gimbal attitude control. *J. Guid. Cont. Dynamics*, 16(4):609–616, 1993.

[41] D. E. Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Trans. Man-Machine Systems*, 10(2):47–53, 1969.

[42] D. E. Whitney. Force feedback control of manipulator fine motions. *Trans. ASME J. Dyn. Systems Meas. Control*, pages 91–97, 1977.

[43] D. E. Whitney and J. L. Nevins. What is the remote centre compliance (RCC) and what can it do? In *Int. Symp. Industrial Robots*, pages 135–152, Washington, DC, 1979.

Index