Robot Kinematics and Dynamics

Herman Bruyninckx
Katholieke Universiteit Leuven
Department of Mechanical Engineering
Leuven, Belgium

Contents

Chapter 1

Introduction

This book explains the *physical* and *mathematical models* that are used to describe, analyse and synthesize *computer-controlled robot motion*. The *specification* of motions for any kind of robot mechanism is within the scope of this book, but the *control* of those specified motions is not part.

Any intelligent robot system robot consists of a combination of software and hardware, Fig. 1.1, with which it has to execute a given task in its current environment, Undoubtedly, *motion* is one of the most important goals of the robot. The different components in the above-mentioned figure have the following features:

- *Mechanism.* Each robot has a mechanical structure, that allows it to move its arms, legs and tools, as well as itself.
- *Sensing.* The robot users its sensors to get set of analog and digital signals that contain information about itself, as well as about what is going on in its environment.
- *Modelling.* The human provides the robot with a set of models, which contain the *knowledge* that the robot needs to understand its own task, to interpret the world around it as well as its own actions in this world. The models are mostly software representations of the physical world, upto an appropriate level of modelling detail; however, some parts of the models can exist in hardware too, for example, artificial markers that are placed in the environment (by the human or by the robot) and that represent a known, modelled part of that environment.
- *Perception.* The raw sensor information is *interpreted*, in software, in the context of these models, resulting in an "understanding" of what is going on around it.
- *Learning.* The sensor information can be used to learn parts of the models that were uncertain before, or even to create new models from experience.
- *Planning.* Since the robot has a model describing the goals of its task, and it has perceived its environment and its own state in that environment, it can then plan its own actions over a certain interval of time in the future, in order to get closer to its task goals.
- *Adapation.* The robot should be able to adapt, on-line, its planned actions to the immediate information about the world that its sensors provide.
- *Control.* On the basis of its task, the perceived status of the world, the robot uses the *instantaneously* collected sensor information to decide how it has to move itself and/or its tools in the immediate future.
- *Actuation.* The output of the control algorithm is converted to appropriate signals that can drive the motors on the mechanism.

1.1 Robot motion

This book focuses on *motion* tasks, and this Section explains the terminology used above. This text deals with the *physical* and *mathematical models* that are used to describe, analyse and synthesize *computer-controlled robot motion*:

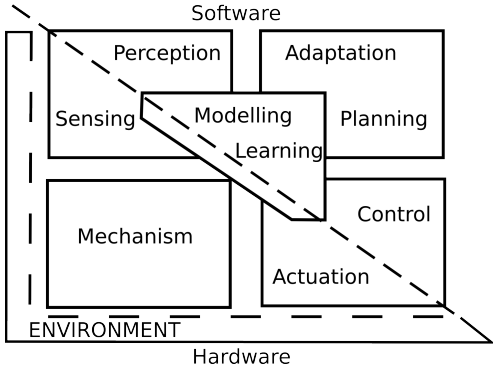

Figure 1.1: A minimal set of components in any intelligent robot.

- **Robots** are in the first place *positioning devices*, that consists of a *mechanical structure*—the so-called *kinematic chain* of the robot—that connects *links* and *joints*, and that carries *motors* (*actuators*) in some or all of the joints, in order to make the mechanical structure move, and *position sensors* on the joints, in order to measure the joints' motion. The goal of the robot motion is to position one or more "tools" and/or sensors on the robot at desired places in the robot's workspace. ("End-effector" is the generic name with which this book will refer those tool and sensor locations on the robot's kinematic chain.)

- **Model**. Every model of the reality is an *approximation* of that reality. This book uses *lumped-parameter* models:

  - each link is an ideal *rigid body* whose mass can be considered to be located ("lumped") in one particular point with inertial properties represented by a 6×6 *mass matrix*.
  - each joint is a perfect mechanical constraint on the motion freedom of the links it interconnects. So, it has no mechanical play, and no backlash if the joint contains a gear. (This is most often the case in industrial robot arms.)
  - all *flexibility* in the robot is located in ideal springs whose elastic properties are fully described via a 6×6 *stiffness matrix*—for the flexibility "lumped" at particular points on a rigid body—or via a scalar *joint stiffness*—for the flexibility "lumped" into a joint about or along its motion axis. The latter flexibility model assumes the joint has only one degree of freedom, which is most often the case in robots; of course, joints with more degrees of freedom exist, but they are more complex to built accurately and to equip with motors and sensors.
  - all *damping* in the robot is located in the joints, whose damping ("friction") properties are fully described by a *friction function* depending on the joint's position and speed. This function can be as simple as a constant scalar, but many more complex, nonlinear friction models have been described in the literature.
  - all positions, velocities, accelerations, forces and torques are measured perfectly, without noise or systematic error, and are the measurements of one uniquely identified point of the robot.

  A *physical* model of a robot makes use of concepts such as: rigid bodies, ideal motion constraints, mass, stiffness, etc. The same reality can be modeled by more than one physical model, depending on the level of detail that the (human) modeler wants to take into account. For example, one model assumes a perfect joint axis, with known angular inertia and Coulomb friction coefficients; another model replaces this simple joint axis by an explicit model of the gear train, including elasticity and backlash at the level of the gears' teeth.

  Each physical model, in turn, can be *represented* by various *mathematical* models: each mathematical model adds a *coordinate representation* to the physical model, i.e., it chooses *one particular way to assign numbers* to the physical model components. For example, the *centre of mass* of a rigid body is a physical concept; the mass of the body, as well as the position of the centre of mass, can be represented by numbers: one number m that gives the mass in *kilograms*, and three other numbers x, y, z, that give the position of the centre of mass with respect to a chosen reference frame. Many other mathematical models are possible for this same physical concept: one could use *grams* or *pounds* to represent the mass, and one could choose a

right-handed or a left-handed reference frame, or polar coordinates, to represent the position of the centre of mass.

- **Motion** of the robot will often be interpreted broadly in this book, meaning both of the following:

  1. the *position*, *velocity* and/or *acceleration* of all joints, and of all links and end-effectors, as functions of time.
  2. the *forces*—on all joint axes and on all links (and tools) of the robot—that cause the motion.

  The context will typically be sufficiently clear to tell the reader whether "motion" means nothing more than, say, only position and velocity, instead of the full meaning described above.

  This book only describes and analyzes *instantaneous* motion, i.e., the current position, velocity and acceleration, and those of immediately preceding or following time instants. So, none of the following is dealt with:

  – motion planning over "longer" time spans.
  – obstacle and collision detection.
  – the sensor processing and decision making about when to change the currently ongoing motion controler, and to switch to another one.

  This text also discusses the *statics* of serial manipulators, i.e., the static equilibrium between forces on the robot's end-effector(s) and forces on its joint axes. The background needed to understand statics is only a small extension to what is needed to understand motion, because of the *duality* properties between the mathematical descriptions of forces and velocities. This integrated analysis of kinematics and statics for rigid bodies and robotic devices is sometimes given the name *kinetostatics*.

- **Computer-controlled**: the robot's computer controls the *force* at its motors, in such a way that the specified motion is realised. The control algorithm has to take into account various, complementary constraints imposed on the robot's motion: (i) the desired motion (*artificial constraints*); (ii) the physical constraints of the kinematic chain (basically, the available motion degrees of freedom, and the saturation of the motors); (iii) the computational constraints of the computing hardware on which the control has to be executed; and (iv) the finite accuracy with which the ongoing motion can be measured.

  The force at the motors is generated, for example, by means of an *electrical motor*, for which the *current* through its coils is the controled physical parameter. Or by means of a *hydraulic motor*, for which the *flow rate* of the oil is the controled physical parameter. Of course, a variety of other actuation mechanisms exists. In practice, the speed of control of (especially electrical) motors is often considered to be infinitely fast (in comparison to the dynamics of the robot mechanics and/or the robot's motion task) such that a control based on *joint velocity* is appropriate; such a control *model* is much simpler than a model that takes the full dynamics of the robot into account.

1.2 Kinematic chain topologies

The **mechanical structures** considered in this text fall into one of the following *topologies*:

- **Serial**: the links and joints are connected in series, Figs. 1.2–1.3.
- **Tree**: the links and joints form a tree, Fig. 1.4.
- **Graph**: the links and joints form a graph, i.e., a structure that has one or more *loops*.

The Graph category contains many special cases, for which customized solution approaches have been developed; for example, **gantry** robots (Fig. 1.6), **parallel** robots (Fig. 1.7), *mobile* robots, or *musculoskeletal* human models. Some robotic devices change their topology during the execution of their task; for example, a walking robot has one or more mechanical loops through the feet it has on the ground; similarly for a robotic hand, that can make loops through the contacts with a manipulated object.

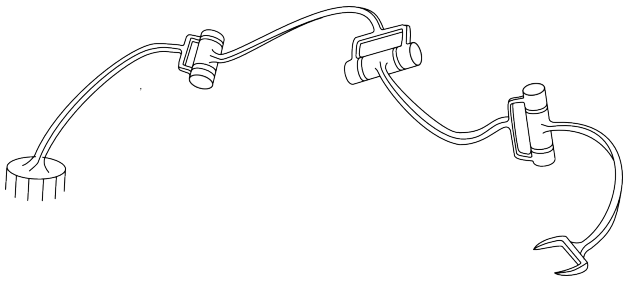

Figure 1.2: Generic serial chain.

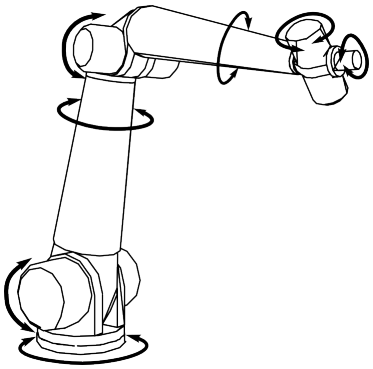

Figure 1.3: Serial robot with seven rotational degrees of freedom.

1.3 Goals

The **goal** of studying the kinematic and dynamic models of robot structures is to understand—for each of the above-mentioned families of kinematic chains—how the following algorithms work, and how they can be applied *most efficiently* to specify, simulate and control robot motion:

- **Forward Kinematics** ("FK", also called "Direct Kinematics): calculate the end-effector motion that results from given joint motions.
- **Inverse Kinematics** ("IK"): calculate the joint motion required to generate a desired end-effector motion.
- **Forward Dynamics** ("FD"): calculate the end-effector motion that results from given forces at the joints.
- **Inverse Dynamics** ("ID"): calculate the joint forces required to generate a desired end-effector motion (possibly together with desired reaction forces against physical constraints acting on the robot).

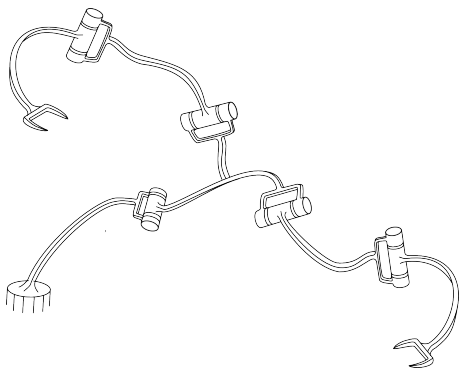

Figure 1.4: Generic tree chain. (Note: a drawing of a humanoid robot kinematic chain would be more appropriate here!)

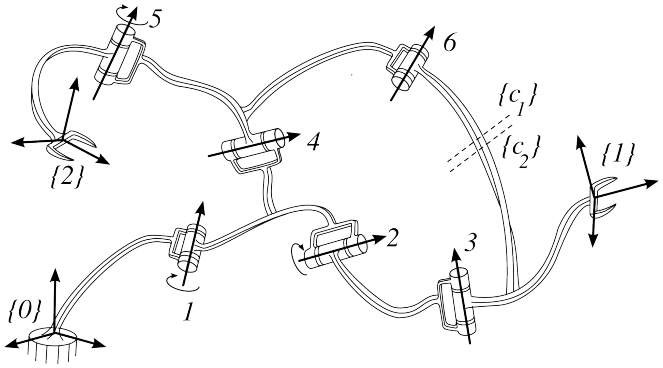

Figure 1.5: Generic graph chain.

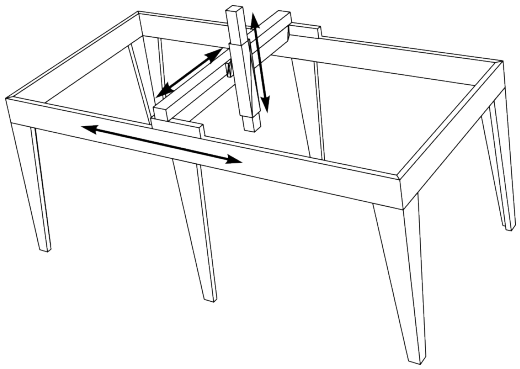

Figure 1.6: Gantry robot, a combination of a parallel and a serial part.

All these algorithms "just" *transform* motion between *joint space* and *Cartesian space*, but, in general, these algorithms can be quite complex and computationally involved. The literature describes a large variety of algorithms, and this variety is motivated by a set of (sometimes conflicting) design requirements:

- *Efficiency for generic structures*: most of the work during the last 50 years has been spent on developing numeric algorithms that work for any possible kinematic chain. The most efficient algorithms have an $\mathcal{O}(n)$ ("*order n*") *complexity*, meaning that the amount of computations is proportional to the number n of joints in the chain.

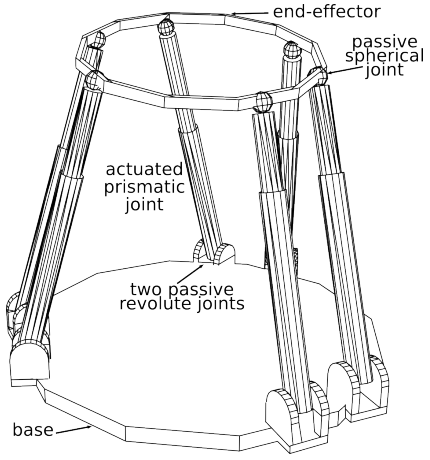

Figure 1.7: One of the generic parallel robot design families, with six "legs", each having one actuated linear joint.

- *Efficiency for one particular kinematic family*: most robots in daily practice do not have a generic kinematic structure, but belong to a specific *kinematic family* with special geometric structure, such as subsequent parallel or orthogonal joint axes. These geometric properties often allow for more efficient algorithms, that are only valid for one particular kinematic family.

  In fact, two kinematic chains are classified into the same kinematic family *if* they can share the same kinematics and dynamics algorithms. (Up to parameters such as link lengths and mass distribution, which in general do not affect the algorithms' design or computational complexity.)

- *Efficient use of parallel computing hardware*: some algorithms lend themselves to be executed in parallel, such that a gain in computation *time* (*not* in number of computations!) can be achieved. The best algorithms realize an $\mathcal{O}(\log(n))$ complexity.

- *Robustness against singularities*: the transformations between joint motion and end-effector motion are, in general, non-linear functions of the joint and/or end-effector positions, hence, the inverse transformations typically have *singularities*, i.e., configurations of the kinematic chain where the inverse is not well-defined. To provide good solutions in the case of singularities makes the kinematic and/or dynamic algorithms more complex and more costly.

1.4 Practical aspects

This book mainly considers *ideal* systems, and mostly *instantaneous* motion, but the reader should be aware of typical practical problems that occur in most systems, and of other algorithms that most robot systems require in addition to the kinematics and dynamics:

- *Friction* can be dominant with respect to inertia, especially for robots with high gear ratios at the joints, and moving at low speed. The problem with friction is that it is so difficult to model accurately, and to identify the model parameters that are different in each particular robot. Often, friction properties also change with temperature, humidity, and wear.

- Elasticity (i.e., non-infinite stiffness) will cause oscillations and inaccuracies in the performed motion. Especially when the robot motion has a high dynamic range, by design (e.g., when trying to make the robot move as fast as possible) as well as by accident (e.g., when making sudden changes in motion, due to impacts or suddenly changing target positions).

- Describing instantaneous motion is seldom enough for real applications, since those require the connection of many instantaneous motions into *tasks* or *skills*, with an appropriate scheduling and decision making based on the information extracted from the robot's sensors.

- Singularities: as mentioned before, the transformations between joint space and Cartesian space sometimes break down. Or rather, the simplest approaches to these transformations do.

- Robot control often requires *estimation* of some directly non-measurable model parameters on the basis of uncertain measurements and models that relate those variables to the measurable ones.

- Underactuation: some joints provide motion degrees of freedom but have no motor to control their motion. This requires more complex motion planning and control.

Chapter 2

Points, vectors, lines and screws

This Chapter describes the *geometrical primitives*—as well as their *coordinate representations*—that are needed in the physical models and the mathematical representations of rigid body kinematics and dynamics: points, vectors, lines, and screws. Most textbooks on classical mechanics deal only with the physical properties of *points* and point masses, and their simple *vector* (coordinate) representations. The study of rigid body mechanics, however, requires more complex primitives: *frames* and *lines* (e.g., to model the position, orientation and axis directions of the revolute or prismatic joints most robots are constructed with), and eventually *screws*, to generalise the line concept into a description of both translation and orientation components.

Later Chapters apply the concepts of this Chapter to the kinematics and dynamics of *one single* rigid body, and of *kinematic chains* of multiple, interconnected rigid bodies.

2.1 Points

Points are the simplest geometric entities. A point's position $\boldsymbol{p}$ can be represented numerically by a 3×1 coordinate "vector":

$$\boldsymbol{p} = \begin{pmatrix} \boldsymbol{p}_x \\ \boldsymbol{p}_y \\ \boldsymbol{p}_z \end{pmatrix}. \tag{2.1}$$

The symbol $\boldsymbol{p}$ is overloaded, denoting: (i) the position of a point without referring to any reference frame in particular, (ii) the position of a point with respect to some identified reference frame $\{f\}$ (indicated by a leading subscript: ${}_f\boldsymbol{p}$), and (iii) the *coordinates* of the point with respect to such a *reference frame* (indicated by trailing subscripts: $\boldsymbol{p}_x$, $\boldsymbol{p}_y$, and $\boldsymbol{p}_z$ for a right-handed reference frame with X, Y and Z axes).

Distance. The distance $d(\boldsymbol{p}^1, \boldsymbol{p}^2)$ between two points $\boldsymbol{p}^1$ and $\boldsymbol{p}^2$ is given by the symmetric *Euclidean distance* function:

$$d(\boldsymbol{p}^1, \boldsymbol{p}^2) = d(\boldsymbol{p}^2, \boldsymbol{p}^1) = (\boldsymbol{p}^1_x - \boldsymbol{p}^2_x)^2 + (\boldsymbol{p}^1_y - \boldsymbol{p}^2_y)^2 + (\boldsymbol{p}^1_z - \boldsymbol{p}^2_z)^2. \tag{2.2}$$

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.

2.2 Vectors

The *geometric* concept of a "vector" is the directed difference between two points. For example, $\boldsymbol{v}^{p,q}$ denotes the vector pointing from the point $\boldsymbol{p}$ to the point $\boldsymbol{q}$. Its *coordinates* can be easily found from the coordinates of both points (of course, all expressed with respect to the same reference frame $\{f\}$), and, as mentioned above, are often denoted by the same symbol:

$$\boldsymbol{v}^{p,q} = \begin{pmatrix} \boldsymbol{v}^{p,q}_x \\ \boldsymbol{v}^{p,q}_y \\ \boldsymbol{v}^{p,q}_z \end{pmatrix} = \begin{pmatrix} \boldsymbol{q}_x \\ \boldsymbol{q}_y \\ \boldsymbol{q}_z \end{pmatrix} - \begin{pmatrix} \boldsymbol{p}_x \\ \boldsymbol{p}_y \\ \boldsymbol{p}_z \end{pmatrix}. \tag{2.3}$$

Although both have a similar coordinate representation, there is a fundamental geometric difference between vectors and points: the addition of vectors is frame-independent (since it depends on the relative positions of the points that make up the vectors being added), but the "addition" of points is not (since it depends on the position of the origin of the reference frame with respect to which the points' coordinates are taken). For example: adding two velocity vectors of the same moving point mass has physical meaning; adding two positions has no such meaning.

Distance. In general, the distance between two vectors is not a well-defined geometric property.

2.3 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.

Distance. 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:

$$d(\boldsymbol{e}^1, \boldsymbol{e}^2) = \arccos(\boldsymbol{e}^1 \cdot \boldsymbol{e}^2), \tag{2.4}$$

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.

2.4 Polar and axial vectors

Two interpretations exist for direction vectors in three-dimensional space, depending on how they incorporate the notion of *orientation*, [34, 84]:

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. As many (but not all) other textbooks, this book *implicitly* uses right-handed reference frames only, but no physical arguments prevent the use of left-handed frames.

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.

The difference between polar and axial vectors is in fact deeper than what was mentioned above: a polar vector isn't a vector at all, but it really is an *operator* on axial vectors in the three-dimensional Euclidean space. The effects of such an operator $\boldsymbol{m}$ are mathematically represented by a 3×3 *skew-symmetric* matrix

$$[\boldsymbol{m}] = \begin{pmatrix} 0 & -m_z & m_y \\ m_z & 0 & -m_x \\ -m_y & m_x & 0 \end{pmatrix} \tag{2.5}$$

to be multiplied with the three-dimensional coordinate vectors of the axial vector on which the operator works, [169, 173]. On other words, premultiplying an axial three-vector with this matrix corresponds to taking the cross

product with the polar vector corresponding to the matrix; and it is well-known that the sign of a cross product depends on the orientation of the world reference frame.

Economy of notation is the only reason why one most often uses a three-vector instead of the 3×3 matrix; it's an (un)fortunate coincidence that the number of independent components in an antisymmetric $n \times n$ matrix is equal to the number of independent components in an $n \times 1$ vector (only!) for $n = 3$: $n = 1/2(n-1)n$. This text will often use the matrix notation "$[\boldsymbol{m}]$" instead of the vector notation "$\boldsymbol{m}\times$" in order to emphasize the fact that a polar vector is actually not a vector.

2.5 Lines–Planes

Lines are very important in robotics because:

- They model ideal one-dimensional *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.

The definitions above cover only lines and planes with an *infinite* extension. One does need more parameters to represent finite *line segments*, or *planar polyhedrals*.

The space of all lines in the three-dimensional Euclidean space is *four-dimensional*. One constructive way to construct this space is by drawing lines from each point in a plane through each point in another, parallel plane.[1] This means that four parameters uniquely define a line: the two coordinates of the first point in the first plane, plus the two coordinates of the second point in the second plane.

The space of all planes in the three-dimensional Euclidean space is *three-dimensional*. One constructive way to construct this space is by drawing the plane through each point in the Euclidean space which is perpendicular to the vector that connects that point to the (arbitrarily chosen) origin of the space.[2] So, the number of planes and points is the same.

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) *direction* of the joint's motion.

2.5.1 Non-minimal two-vector coordinates

This Section introduces one particular representation of a (directed) line (others will follow in later Sections), denoted by $\mathcal{L}(\boldsymbol{p}, \boldsymbol{d})$, which defines the line $\mathcal{L}$ by an ordered set of two vectors: (i) one point vector $\boldsymbol{p}$ (indicating the position of an arbitrary point on $\mathcal{L}$ with respect to a known reference frame), and (ii) one free direction vector $\boldsymbol{d}$, giving the line a direction. (The *direction* of the line is not really 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. Hence, there exist two constraints between the two vectors defining the line:

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 the following six-dimensional coordinate vector:

$$\boldsymbol{l} = (\boldsymbol{p}_x \; \boldsymbol{p}_y \; \boldsymbol{p}_z \; \boldsymbol{d}_x \; \boldsymbol{d}_y \; \boldsymbol{d}_z)^T. \tag{2.6}$$

The order of the two vectors in this representation is completely arbitrary.

[1] This construction has problems with the lines that are parallel to both planes...

[2] This construction has problems for the planes through that origin...

2.5.2 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*, [24, 39, 121, 205, 204]. This representation was finally named after Plücker. We denote this Plücker representation by $\mathcal{L}_{\text{pl}}(\boldsymbol{d}, \boldsymbol{m})$. 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, $\boldsymbol{m} = \boldsymbol{p} \times \boldsymbol{d}$. $\boldsymbol{m}$ is independent of which point $\boldsymbol{p}$ on the line is chosen: $\boldsymbol{p} \times \boldsymbol{d} = (\boldsymbol{p} + t\boldsymbol{d}) \times \boldsymbol{d}$, for any value of t. $\boldsymbol{p}$ is an *axial* direction vector, and $\boldsymbol{d}$ is a *polar* vector, i.e., the moment of the direction vector about the origin of the *chosen* reference frame. And the two vectors $\boldsymbol{d}$ and $\boldsymbol{m}$ are always *orthogonal*:

$$\boldsymbol{d} \cdot \boldsymbol{m} = 0. \tag{2.7}$$

The advantage of the Plücker coordinates is that they are *homogeneous*: $\mathcal{L}_{\text{pl}}(k\boldsymbol{d}, k\boldsymbol{m}), k \in \mathbb{R}$, 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. 2.6.) A coordinate representation of the line in Plücker coordinates is the following six-vector $\boldsymbol{l}$:

$$\boldsymbol{l} = (\boldsymbol{d}_x \ \boldsymbol{d}_y \ \boldsymbol{d}_z \ \boldsymbol{m}_x \ \boldsymbol{m}_y \ \boldsymbol{m}_z)^T. \tag{2.8}$$

with $\boldsymbol{d}$ and $\boldsymbol{m}$ the three-dimensional (coordinate) vectors representing the direction and moment vectors, respectively. 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 Eq. (2.7).

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., $\boldsymbol{d} \cdot \boldsymbol{p} = 0$. Since $\boldsymbol{d} \cdot \boldsymbol{d}$ is a *scalar*, while $\boldsymbol{d} \times \boldsymbol{m}$ is a vector, we find $\boldsymbol{p}$ as follows:

$$\boldsymbol{p} = \frac{\boldsymbol{d} \times \boldsymbol{m}}{\boldsymbol{d} \cdot \boldsymbol{d}}. \tag{2.9}$$

Intersection of lines

Plücker coordinates allow a simple test to see whether two (non-parallel) lines $\boldsymbol{l}^1$ and $\boldsymbol{l}^2$ intersect. (This test comes in handy when applied to testing the orthogonality of robot joint axes.) The lines intersect, if and only if, (Fig. 2.1),

$$\boldsymbol{d}^1 \cdot \boldsymbol{m}^2 + \boldsymbol{d}^2 \cdot \boldsymbol{m}^1 = 0. \tag{2.10}$$

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 $\boldsymbol{l}^1$ to a point $\boldsymbol{r}^2$ on $\boldsymbol{l}^2$ lies in this plane. In other words, it is orthogonal to the common normal direction, given by $\boldsymbol{d}^1 \times \boldsymbol{d}^2$: $0 = (\boldsymbol{d}^1 \times \boldsymbol{d}^2) \cdot (\boldsymbol{r}^2 - \boldsymbol{r}^1) = (\boldsymbol{d}^1 \times \boldsymbol{d}^2) \cdot \boldsymbol{r}^2 - (\boldsymbol{d}^1 \times \boldsymbol{d}^2) \cdot \boldsymbol{r}^1 = -\boldsymbol{d}^1 \cdot (\boldsymbol{d}^2 \times \boldsymbol{r}^2) - (\boldsymbol{r}^1 \times \boldsymbol{d}^1) \cdot \boldsymbol{d}^2 = -\boldsymbol{d}^1 \cdot \boldsymbol{m}^2 - \boldsymbol{d}^2 \cdot \boldsymbol{m}^1$.

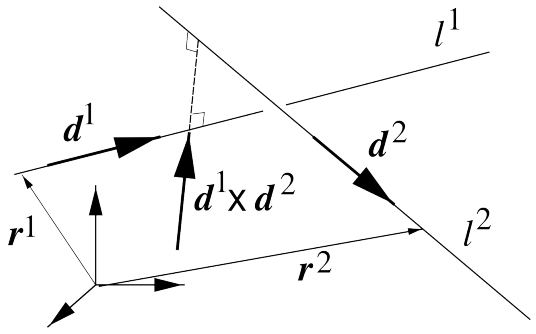

Figure 2.1: Two crossing lines $\boldsymbol{l}^1$ and $\boldsymbol{l}^2$, with common normal direction $\boldsymbol{d}^1 \times \boldsymbol{d}^2$. $\boldsymbol{r}^1$ and $\boldsymbol{r}^2$ denote points on the lines, with respect to the origin of the reference frame.

Common normal. The common normal line $\boldsymbol{l}^{cn}$ to two lines can be calculated as follows. Figure 2.1 shows that the following vector closure relation holds:

$$\boldsymbol{r}^1 + k\boldsymbol{d}^1 + l\boldsymbol{d}^1 \times \boldsymbol{d}^2 + m\boldsymbol{d}^2 = \boldsymbol{r}^2. \tag{2.11}$$

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 (2.11) can be written as a set of three linear equations in the three unknowns k, l, and m:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{2.12}$$

with $\boldsymbol{A} = \left(\boldsymbol{d}^1 \ (\boldsymbol{d}^1 \times \boldsymbol{d}^2) \ \boldsymbol{d}^2\right), \boldsymbol{x} = (k \ l \ m)^T$, 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 $\boldsymbol{l}^{cn}$ are

$$\boldsymbol{l}^{cn} = \begin{pmatrix} \boldsymbol{d}^1 \times \boldsymbol{d}^2 \\ \boldsymbol{p} \times (\boldsymbol{d}^1 \times \boldsymbol{d}^2) \end{pmatrix}, \quad \text{with} \quad \boldsymbol{p} = \boldsymbol{r}^1 + k\boldsymbol{d}^1. \tag{2.13}$$

2.5.3 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. 2.2), [65, 103]. 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.

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*, Fig. 2.4.

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 any singularities in the whole space. This singularity problem can be approached in two ways:

1. Using more than one so-called *coordinate patch*, i.e., a selection of such patches for which no singularities exist for the chosen representation. Of course, appropriate transformations have to be executed whenever the line "moves" from one patch to the other.

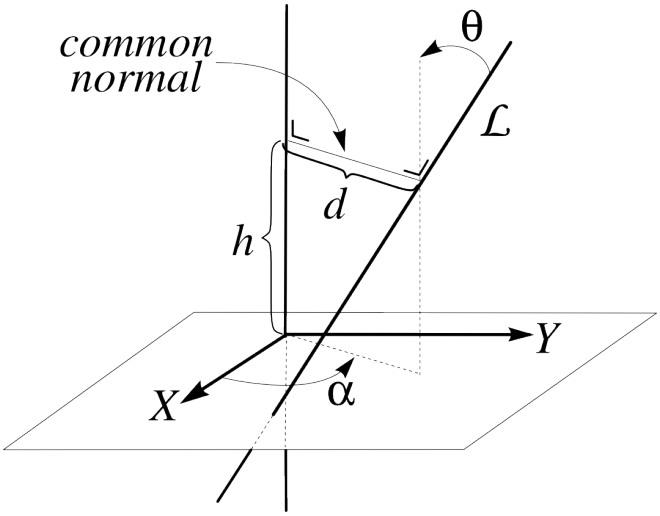

Figure 2.2: Line parameters in the Denavit-Hartenberg convention.

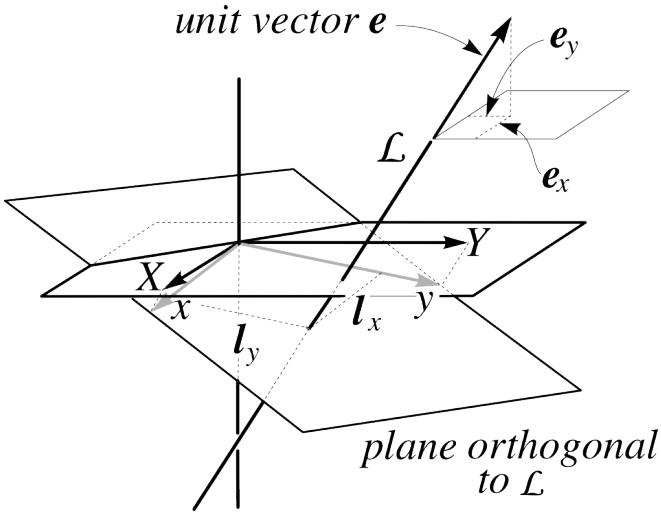

Figure 2.3: Line parameters in the Hayati-Roberts convention.

Figure still to be drawn...

Figure 2.4: Singular configuration for the Denavit-Hartenberg line representation convention.

To understand the concept of "coordinate patch", it might be good to look at the example of atlases of the earth: 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, [231, 258, 293]. 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 world frame, its orientation, and the positive directions of distances and angles. This problem can show up with any representation (minimal or not) but minimal representations suffer most.

The (dis)advantages of DH parameters play an important role in the *calibration* of a robot's geometrical model, i.e., the measurement process in which one wants to find out the exact relative position and orientation of the robot's links and joints, at least within the measurement accuracy of the sensors. Indeed, as with any man-made device, a robot's real geometry can differ from the nominal geometric model that came from the designers' drawing board, 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, on the work floor. 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, [19]. 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 cumbersome.

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 sensor, a distance sensor, or a 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.

2.5.4 Hayati-Roberts coordinates

Another minimal line representation is the *Hayati-Roberts* line representation, that we denote by $\mathcal{L}_{hr}(\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{l}_x, \boldsymbol{l}_y)$, [19, 106, 217] (Fig. 2.3):

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.

2.5.5 Line coordinate transformations

This Section explains how to transform between the various coordinate representations introduced in the previous Sections. (TODO!)

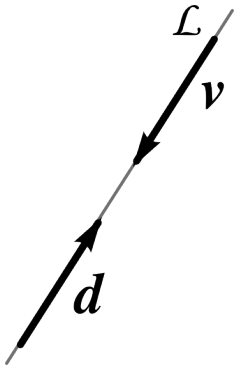

Figure 2.5: Components of a screw: the line $\mathcal{L}$, and the two vectors $\boldsymbol{d}$ and $\boldsymbol{v}$ bound to this line.

2.6 Screws

The geometrical concept of a screw, Fig. 2.5, is an extension of the line concept: a screw is a line with *two* vectors on that line. An alternative definition is: a line plus *one* vector bound to lie on that line, since the line itself already contains a natural direction vector. This screw concept will turn out very appropriate to represent mathematically the physical concepts of displacement and velocity of a rigid body, and of force on a rigid body, see Sect. 4.1.7 and following.

Recall from Sect. 2.5.2 that the Plücker representation of a line is $\mathcal{L}_{\text{pl}}(\boldsymbol{d}, \boldsymbol{m})$, with $\boldsymbol{m}$ the moment of the direction vector $\boldsymbol{d}$ of the line, with respect to the (arbitrarily chosen) origin: $\boldsymbol{m} = \boldsymbol{p} \times \boldsymbol{d}$, where $\boldsymbol{p}$ is the position vector of *a* point on the line. One now adds a vector $\boldsymbol{v}$ (parallel to the line) to this moment vector $\boldsymbol{m}$: $\boldsymbol{m} = \boldsymbol{p} \times \boldsymbol{d} + \boldsymbol{v}$. Such an "extended" line is called a *screw*, [5, 15, 24, 121, 290], and is denoted by $\mathcal{L}_{\text{sc}}(\boldsymbol{d}, \boldsymbol{m})$. The line of the screw is called the *screw axis*. The ratio between the parallel vectors $\boldsymbol{d}$ (the direction vector of the screw) and $\boldsymbol{m}$ is called the *pitch* p of the screw:

$$p\, \boldsymbol{d} = \boldsymbol{m}. \tag{2.14}$$

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 units of *length*.

Recall that a *line* representation has two degrees-of-freedom unspecified. 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 $\boldsymbol{s}$ and $k\boldsymbol{s}$ lie on the same line and have the same pitch, for any scalar number k. The above-mentioned homogeneity 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 $\boldsymbol{d}$ and $\boldsymbol{m}$ are determined by this speed. Such a screw with a given speed is called a *twist* (or "generalized velocity"). Sir William Rowan Hamilton (1788–1856) [99] called it a "screw with a magnitude." William Kingdon Clifford (1845–1879) called it a *motor*, [50, 265, 280, 286], this word being the contraction of "motion" and "vector."

2.6.1 Screw coordinates—Screw axis

Once a world reference frame is chosen, the *coordinates* of the screw $\mathcal{L}_{\text{sc}}(\boldsymbol{d}, \boldsymbol{m})$ form a 6×1 coordinate vector $\mathbf{s}$:

$$\mathbf{s} = \begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{m} \end{pmatrix}. \tag{2.15}$$

Similarly to Eq. (2.9), it is straightforward to find the position vector $\boldsymbol{p}$ of the point on the line *closest to the origin*:

$$\boldsymbol{p} = \frac{\boldsymbol{d} \times \boldsymbol{m}}{\boldsymbol{d} \cdot \boldsymbol{d}}. \tag{2.16}$$

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

$$p = \frac{\boldsymbol{d} \cdot \boldsymbol{m}}{\boldsymbol{d} \cdot \boldsymbol{d}}. \tag{2.17}$$

So, the rather simple calculations above find a *screw axis* when given the six numbers in the screw's coordinate vector.

2.6.2 Reciprocity of screws

Two screws $\mathbf{s}_1$ and $\mathbf{s}_2$—with coordinate column vectors $\mathbf{s}_1 = (\boldsymbol{d}_1^T \, \boldsymbol{m}_1^T)^T$ and $\mathbf{s}_2 = (\boldsymbol{d}_2^T \, \boldsymbol{m}_2^T)^T$—are *reciprocal* if

$$\mathbf{s}_1^T \boldsymbol{\Delta} \mathbf{s}_2 = \boldsymbol{d}_1^T \boldsymbol{m}_2 + \boldsymbol{d}_2^T \boldsymbol{m}_1 = 0, \tag{2.18}$$

with

$$\boldsymbol{\Delta} = \begin{pmatrix} \mathbf{0}_3 & \boldsymbol{I}_3 \\ \boldsymbol{I}_3 & \mathbf{0}_3 \end{pmatrix}. \tag{2.19}$$

The mapping from two screws $\mathbf{s}_1$ and $\mathbf{s}_2$ into the scalar number $\mathbf{s}_1^T \boldsymbol{\Delta} \mathbf{s}_2 = \boldsymbol{d}_1^T \boldsymbol{m}_2 + \boldsymbol{d}_2^T \boldsymbol{m}_1$ is called the *reciprocal product* of both screws. This mapping also allows to find the *reciprocal set* of a screw $\mathbf{s}_1$, i.e., all screws $\mathbf{s}_2$ whose reciprocal product with $\mathbf{s}_1$ vanishes.

From a numerical point of view, the calculation of this reciprocal set looks a lot like the calculation of an orthogonal complement. Indeed, the numerical Gram-Schmidt procedure, [85],[3] is applicable, if one first premultiplies all screws in $\{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ by the matrix $\boldsymbol{\Delta}$ of Eq. (2.19).

2.6.3 Screws as elements of a vector space

It was mentioned already before that 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" "$\cdot$") of two vectors is a well-defined scalar (e.g., force times displacement is energy), and (iii) the *vector product* (or "cross product" "$\times$") 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 typical textbooks usually do not treat concepts and properties of *screws*, but screws

[3] More original reference needed!

have the three above-mentioned vector properties too, with the *spatial scalar product*, or "pairing," replacing the three-vector scalar product, and the *motor product*, [25, 265, 280], (also called the *spatial cross product*, [76], replacing the three-vector cross product (and denoted by the same symbol "×"). Brand [25] proved that, in terms of this three-vector cross product, the motor product is given by

$$\mathbf{s}^1 \times \mathbf{s}^2 = \begin{pmatrix} \boldsymbol{d}^1 \times \boldsymbol{d}^2 \\ \boldsymbol{m}^1 \times \boldsymbol{d}^2 - \boldsymbol{m}^2 \times \boldsymbol{d}^1 \end{pmatrix}. \tag{2.20}$$

2.7 Transformation of coordinate representations

This Section describes how the coordinate representations of the objects introduced in this and previous Chapters transform under a change of reference frame. Two reference changes are relevant: (i) a change of the world reference frame, or (ii) a change of a rigid body reference frame, Fig. 2.6.

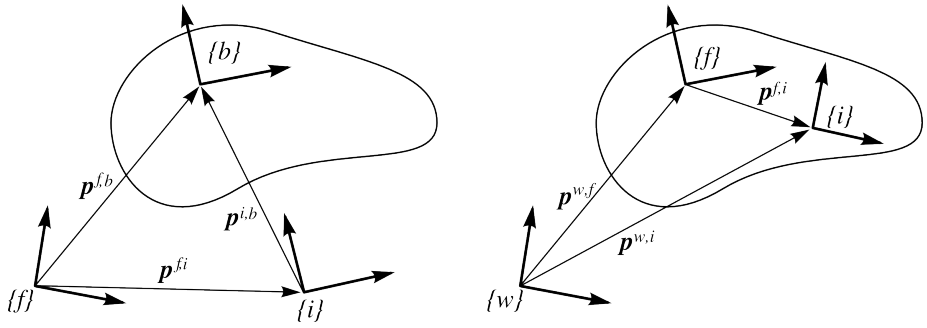

Figure 2.6: Transformations of screw-like geometric entities under a change of world reference frame (left) and body-fixed reference frames (right).

2.7.1 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

$${}_f\boldsymbol{v} = {}^i_f\boldsymbol{R}\ {}_i\boldsymbol{v}. \tag{2.21}$$

Homogeneous point vector components transform as

$$\begin{pmatrix} {}_f\boldsymbol{p} \\ 1 \end{pmatrix} = {}^i_f\boldsymbol{T} \begin{pmatrix} {}_i\boldsymbol{p} \\ 1 \end{pmatrix}. \tag{2.22}$$

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

2.7.2 Line transformation

A line in Plücker coordinates (Sect. **??**) has a representation $\mathcal{L}_{\rm pl}(\boldsymbol{d}, \boldsymbol{m})$. 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. (2.9). 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*:

$${}_f\boldsymbol{d} = {}^i_f\boldsymbol{R}\ {}_i\boldsymbol{d}. \tag{2.23}$$

2. The moment vector $\boldsymbol{m}$ changes (the physical three-vector, as well as its coordinates) if the frames have a *different origin*:

$$\begin{aligned} {}_f\boldsymbol{m} &= {}_f\boldsymbol{p}^{f,l} \times {}_f\boldsymbol{d} \\ &= ({}_f\boldsymbol{p}^{f,i} + {}_f\boldsymbol{p}^{i,l}) \times {}_f\boldsymbol{d} \\ &= {}_f\boldsymbol{p}^{f,i} \times ({}^i_f\boldsymbol{R}\ {}_i\boldsymbol{d}) + {}^i_f\boldsymbol{R}\ \left({}_i\boldsymbol{p}^{i,l} \times {}_i\boldsymbol{d}\right) \\ &= \left[{}_f\boldsymbol{p}^{f,i}\right] {}^i_f\boldsymbol{R}\ {}_i\boldsymbol{d} + {}^i_f\boldsymbol{R}\ {}_i\boldsymbol{m}. \end{aligned} \tag{2.24}$$

Combining the transformations (2.23) and (2.24) gives

$$\begin{pmatrix} {}_f\boldsymbol{d} \\ {}_f\boldsymbol{m} \end{pmatrix} = \begin{pmatrix} {}^i_f\boldsymbol{R} & \boldsymbol{0}_3 \\ \left[{}_f\boldsymbol{p}^{f,i}\right] {}^i_f\boldsymbol{R} & {}^i_f\boldsymbol{R} \end{pmatrix} \begin{pmatrix} {}_i\boldsymbol{d} \\ {}_i\boldsymbol{m} \end{pmatrix}. \tag{2.25}$$

Recall that $[\boldsymbol{p}]$ denotes the skew-symmetric matrix that corresponds to taking the vector product with the three-vector $\boldsymbol{p}$, Eq. (3.19).

2.7.3 Screw transformation

A screw $\mathcal{L}_{\text{sc}}(\boldsymbol{d}, \boldsymbol{v})$ consists of two three-vectors bound to a line (Sect. 2.6.1). Its coordinates with respect to a reference frame $\{i\}$ have been represented as, Eq. (2.15),

$${}_i\mathbf{s} = \begin{pmatrix} {}_i\boldsymbol{d} \\ {}_i\boldsymbol{p}^{i,l} \times {}_i\boldsymbol{d} + {}_i\boldsymbol{v} \end{pmatrix},$$

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. (2.23), 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

$$\begin{aligned} {}_f\boldsymbol{m} &= {}_f\boldsymbol{p}^{f,l} \times {}_f\boldsymbol{d} + {}_f\boldsymbol{v} \\ &= ({}_f\boldsymbol{p}^{f,i} + {}_f\boldsymbol{p}^{i,l}) \times {}_f\boldsymbol{d} + {}_f\boldsymbol{v} \\ &= {}_f\boldsymbol{p}^{f,i} \times ({}^i_f\boldsymbol{R}\ {}_i\boldsymbol{d}) + {}^i_f\boldsymbol{R}\left({}_i\boldsymbol{p}^{i,l} \times {}_i\boldsymbol{d} + {}_i\boldsymbol{v}\right). \end{aligned} \tag{2.26}$$

Hence, the transformation matrix of the screw coordinate six-vector turns out to be exactly the same as for the transformation of a line, [120, 196, 213, 283, 290]. This text calls it the *screw transformation matrix* ${}^i_f\boldsymbol{S}$ (or *screw transform* for short):

$${}^i_f\boldsymbol{S} = \begin{pmatrix} {}^i_f\boldsymbol{R} & \boldsymbol{0}_{3\times3} \\ \left[{}_f\boldsymbol{p}^{f,i}\right] {}^i_f\boldsymbol{R} & {}^i_f\boldsymbol{R} \end{pmatrix}. \tag{2.27}$$

(This name is not commonly accepted in the literature...) Two computationally useful properties of the screw transform are that it can be built from the homogeneous transform with only one matrix multiplication, and that it has unit determinant.

2.7.4 Inverse of screw transformation matrix

By definition, the *inverse* of ${}^i_f\boldsymbol{S}$ is given by

$$\left({}^i_f\boldsymbol{S}\right)^{-1} = {}^f_i\boldsymbol{S} = \begin{pmatrix} {}^f_i\boldsymbol{R} & \boldsymbol{0}_3 \\ \left[{}_i\boldsymbol{p}^{i,f}\right] {}^f_i\boldsymbol{R} & {}^f_i\boldsymbol{R} \end{pmatrix}. \tag{2.28}$$

It is easy to check that this is equal to

$$\left({}^i_f\boldsymbol{S}\right)^{-1} = \boldsymbol{\Delta}\ {}^i_f\boldsymbol{S}^T\ \boldsymbol{\Delta}, \tag{2.29}$$

with

$$\boldsymbol{\Delta} = \begin{pmatrix} \boldsymbol{0}_3 & \boldsymbol{I}_3 \\ \boldsymbol{I}_3 & \boldsymbol{0}_3 \end{pmatrix}. \tag{2.30}$$

Hence, calculating the inverse is computationally very efficient.

2.7.5 Spatial transpose

Equation (2.29) is a special case of the so-called *spatial transpose* of a matrix $\boldsymbol{X}$, [76, 120], denoted by $\boldsymbol{X}^S$:

$$\text{if} \quad \boldsymbol{X} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix} \quad \text{then} \quad \boldsymbol{X}^S = \boldsymbol{\Delta} \boldsymbol{X}^T \boldsymbol{\Delta}. = \begin{pmatrix} \boldsymbol{D}^T & \boldsymbol{B}^T \\ \boldsymbol{C}^T & \boldsymbol{A}^T \end{pmatrix}. \tag{2.31}$$

This definition is attractive in the particular case of a 6×6 screw transformation matrix, because its special form makes it *spatially orthogonal*:

$$\boldsymbol{S}^S \, \boldsymbol{S} = \boldsymbol{I}_{6\times 6}. \tag{2.32}$$

This *formally looks like* the (physical) orthogonality of a 3×3 rotation matrix, Sect. 3.2.

2.7.6 Infinitesimal screw 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. (2.27) becomes an *infinitesimal screw transformation matrix* $\boldsymbol{S}_\Delta$, [265, 290]:

$$\boldsymbol{S}_\Delta(\mathbf{t}_\Delta) = \begin{pmatrix} 1 & -\delta_z & \delta_y & 0 & 0 & 0 \\ \delta_z & 1 & -\delta_x & 0 & 0 & 0 \\ -\delta_y & \delta_x & 1 & 0 & 0 & 0 \\ 0 & -d_z & d_y & 1 & -\delta_z & \delta_y \\ d_z & 0 & -d_x & \delta_z & 1 & -\delta_x \\ -d_y & d_x & 0 & -\delta_y & \delta_x & 1 \end{pmatrix}. \tag{2.33}$$

2.7.7 Active and passive screw transformations

The *passive* interpretation of the result of $\boldsymbol{S}$ and $\boldsymbol{S}_\Delta$ on a screw is that it 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.

Chapter 3

Orientation of rigid bodies

This Chapter introduces the physical properties of the concepts of orientation and rotation of a rigid body, and the various mathematical representations that are used in robotics.

"Rotation" or "(relative) orientation" is the main difference between the motion of *points* and the motion of *rigid bodies*: a point has no orientation, while a rigid body has three degrees of motion freedom in orientation. Every mathematical representation with only three parameters inevitably has *coordinate singularities* at a number of orientation configurations; a coordinate singularity occurs whenever a small change in the physical configuration of the represented system (i.c., a small change in orientation) cannot be represented by a small change in coordinates.

The second important fact to take home from this Chapter is that rotational velocity ("angular velocity") is *not* the time derivative of any representation of orientation; however, *integrating factors* always exist. Hence, orientation coordinates are integrable (or "*holonomic*").

3.1 Definition of relative orientation

Imagine a set of *orthogonal* and *right-handed* reference frames, all having the same point as origin (Fig. 3.1). These frames represent different *orientations* of the rigid body to which they are fixed. The orientation of a frame is not an absolute concept, but a relative one, 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.

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.

3.2 Rotation matrix

This Section describes rotation matrices as appropriate and very often used mathematical representations of relative orientation; later Sections introduce alternative representations.

Several coordinate representations exist to express the relative orientation of a frame $\{b\}$ with respect to a frame $\{a\}$. The 3×3 *rotation matrix* ${}^a_b\boldsymbol{R}$ is among the most popular. Other names for this matrix are *orientation matrix*, or *matrix of direction cosines*. 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. 3.2):

$$ {}^b_a\boldsymbol{R} = \left(R_{ij}\right) = \begin{pmatrix} {}_a\boldsymbol{x}^b & {}_a\boldsymbol{y}^b & {}_a\boldsymbol{z}^b \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}^b\cdot\boldsymbol{x}^a & \boldsymbol{y}^b\cdot\boldsymbol{x}^a & \boldsymbol{z}^b\cdot\boldsymbol{x}^a \\ \boldsymbol{x}^b\cdot\boldsymbol{y}^a & \boldsymbol{y}^b\cdot\boldsymbol{y}^a & \boldsymbol{z}^b\cdot\boldsymbol{y}^a \\ \boldsymbol{x}^b\cdot\boldsymbol{z}^a & \boldsymbol{y}^b\cdot\boldsymbol{z}^a & \boldsymbol{z}^b\cdot\boldsymbol{z}^a \end{pmatrix}. \tag{3.1} $$

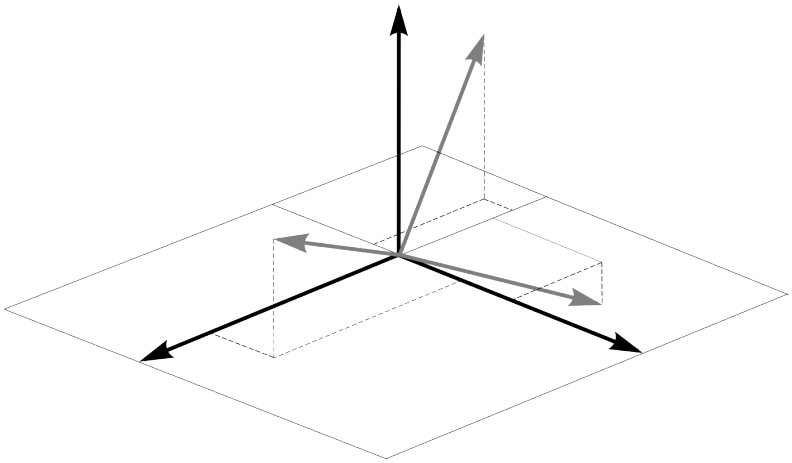

Figure 3.1: Two orthogonal reference frame rotated with respect to each other. (The depicted plane has no specific meaning, and is only used to allow a better 3D interpretation of the 2D drawing.) Fig. 3.2 shows how a coordinate representation of this orientation can be constructed.

${}_a\boldsymbol{x}^b$ 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 $\boldsymbol{z}^a$.

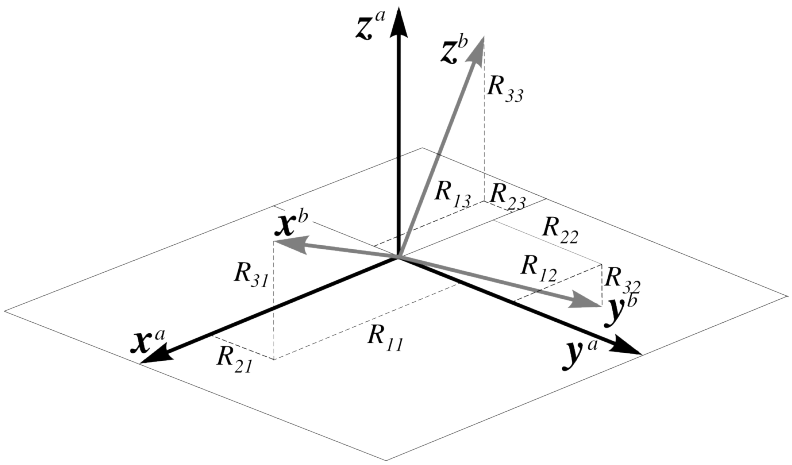

Figure 3.2: Components R_{ij} of a rotation matrix ${}^b_a\boldsymbol{R}$.

From the above-mentioned construction of the rotation matrix, the following fact is obvious, but important: a rotation matrix is a *unique* and *unambiguous* 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.

Many physics, mechanics or geometry books (e.g., [53, 84, 96, 158]) 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.

Equation 3.1 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, in vector-matrix notation:

$$ {}_a\boldsymbol{p} = {}^b_a\boldsymbol{R}\ {}_b\boldsymbol{p}, \quad \text{or} \quad \begin{pmatrix} {}_a\boldsymbol{p}_x \\ {}_a\boldsymbol{p}_y \\ {}_a\boldsymbol{p}_z \end{pmatrix} = {}^b_a\boldsymbol{R} \begin{pmatrix} {}_b\boldsymbol{p}_x \\ {}_b\boldsymbol{p}_y \\ {}_b\boldsymbol{p}_z \end{pmatrix}. \tag{3.2} $$

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."

3.3 noa notation

One sometimes encounters the notation ***noa*** for the three axes of a right-handed orthogonal reference frame. This nomenclature is due to Richard Paul, [198], 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.

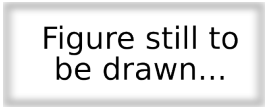

Figure 3.3: Parallel jaw gripper, with noa axes.

3.4 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. 3.4, where $\boldsymbol{p} = \boldsymbol{I}$ 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.

The rotation matrix of this rotation is easily derived from Fig. 3.4:

$$ \boldsymbol{R}(X, \alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix}. \tag{3.3} $$

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

$$ \begin{pmatrix} \boldsymbol{p}'_x \\ \boldsymbol{p}'_y \\ \boldsymbol{p}'_z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_x \\ \boldsymbol{p}_y \\ \boldsymbol{p}_z \end{pmatrix}. \tag{3.4} $$

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:

$$ \boldsymbol{R}(Y, \alpha) = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix}, \qquad \boldsymbol{R}(Z, \alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{3.5} $$

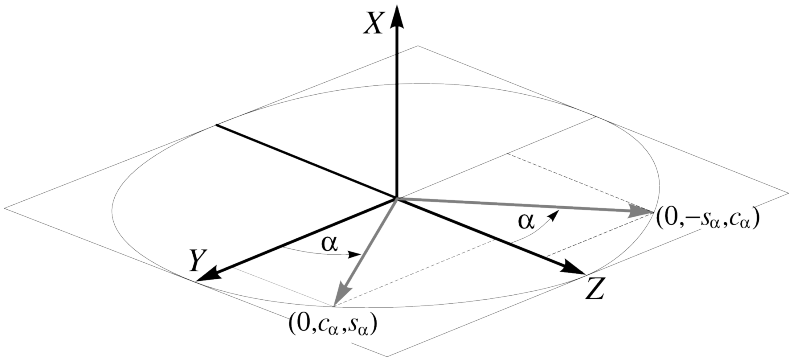

Figure 3.4: Rotation over an angle α about the X axis. c_α stands for $\cos(\alpha)$, and s_α stands for $\sin(\alpha)$.

3.5 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, emphactive interpretation: R *moves* the frame $\{a\}$ into the frame $\{b\}$, [187]. 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:

$$\begin{pmatrix} {}_a\boldsymbol{x}^b & {}_a\boldsymbol{y}^b & {}_a\boldsymbol{z}^b \end{pmatrix} = \boldsymbol{R} \begin{pmatrix} {}_a\boldsymbol{x}^a & {}_a\boldsymbol{y}^a & {}_a\boldsymbol{z}^a \end{pmatrix} = \boldsymbol{R}. \tag{3.6}$$

Equation (3.1) implies that $\boldsymbol{R} = {}^b_a\boldsymbol{R}$, i.e., active and passive interpretations of orientation and rotation have the same matrix representation. While the passive form transforms coordinates of the *same* 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 *another* spatial point (or vector) (i.e., it represents "rotation"). Both interpretations are represented by the same matrix.

3.6 Inverse of a rotation

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 (3.1) gives

$${}^a_b\boldsymbol{R} = \begin{pmatrix} \boldsymbol{x}^a \cdot \boldsymbol{x}^b & \boldsymbol{y}^a \cdot \boldsymbol{x}^b & \boldsymbol{z}^a \cdot \boldsymbol{x}^b \\ \boldsymbol{x}^a \cdot \boldsymbol{y}^b & \boldsymbol{y}^a \cdot \boldsymbol{y}^b & \boldsymbol{z}^a \cdot \boldsymbol{y}^b \\ \boldsymbol{x}^a \cdot \boldsymbol{z}^b & \boldsymbol{y}^a \cdot \boldsymbol{z}^b & \boldsymbol{z}^a \cdot \boldsymbol{z}^b \end{pmatrix}. \tag{3.7}$$

Comparing Eqs (3.1) and (3.7) yields two computationally interesting properties of rotation matrices:

1. the inverse of a rotation matrix is its transpose (which requires no computations to find!):

$${}^a_b\boldsymbol{R} = {}^b_a\boldsymbol{R}^T, \tag{3.8}$$

   or, without reference to specific frames:

$$\boldsymbol{R}^{-1} = \boldsymbol{R}^T. \tag{3.9}$$

2. each rotation matrix is an *orthogonal linear transformation* (i.e., it is a so-called *orthogonal matrix*), since it satisfies the following six orthogonality *constraints*:

$$\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}_3. \tag{3.10}$$

3.7 Non-minimal representation

A direct consequence of Eq. (3.10) is that 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 in Eq. (3.10)

uniquely determine the six dependent parameters. One of the big advantages of such a non-minimal representation is that it has *no coordinate singularities*; this will not be the case for *any* of the minimal representations discussed later.

3.8 Composition of rotations

One of the major properties of rotations is that *composition* of rotations is a *multiplicative* operation, represented, for example, by *multiplication* of rotation matrices. The following paragraphs of this Section will explain this property in full detail.

One of the consequences is that the "sum" or "difference" of two orientations does not have physical sense. So, although it would be *possible* to add or subtract their mathematical representations (such as orientation matrices) this mathematical operation does not correspond to anything physical. Or, in still other words, the addition or subtraction of rotation matrices are not *invariant* operations: the result will typically be different when the operation is performed with respect to different reference frames, or when using different physical units, etc.

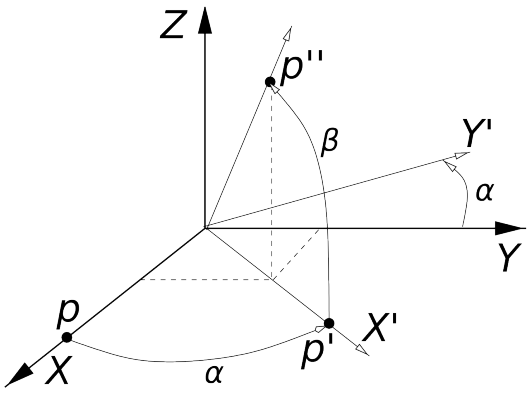

Figure 3.5: Rotation over an angle α about the Z axis, followed by a rotation about the moved Y-axis (i.e., Y') over an angle $-\beta$.

Figure 3.5 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 $\{XYZ\}$:

$$\boldsymbol{p}'' = \begin{pmatrix} c_\alpha c_\beta \\ s_\alpha c_\beta \\ s_\beta \end{pmatrix}, \tag{3.11}$$

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 $-\beta$:

$$\boldsymbol{R}(ZY, \alpha, -\beta) = \begin{pmatrix} c_\alpha c_\beta & -s_\alpha & -c_\alpha s_\beta \\ s_\alpha c_\beta & c_\alpha & -s_\alpha s_\beta \\ s_\beta & 0 & c_\beta \end{pmatrix} = \boldsymbol{R}(Z, \alpha)\boldsymbol{R}(Y, -\beta). \tag{3.12}$$

Somewhere around 1840, [45, 218], the French mathematician Olinde Rodrigues (1794–1851) seems to have been the first to find the coordinate expressions for composing rotations in this way.

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

$$\boldsymbol{R}^{-1}(X, \alpha) = \boldsymbol{R}(X - \alpha). \tag{3.13}$$

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

$$\boldsymbol{R}^{-1}(ZY, \alpha, -\beta) = \boldsymbol{R}(Y, \beta)\boldsymbol{R}(Z, -\alpha). \tag{3.14}$$

3.9 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 $\boldsymbol{p}(t^2)$:

$$\boldsymbol{v} = \frac{d\boldsymbol{p}}{dt} \approx \frac{\boldsymbol{p}(t^2) - \boldsymbol{p}(t^1)}{t^2 - t^1}. \tag{3.15}$$

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:

$$ {}_a\boldsymbol{p}(t) = {}^b_a\boldsymbol{R}(t)\, {}_b\boldsymbol{p}. \tag{3.16}$$

2. Assume that reference frame $\{b\}$ in Eq. (3.2) 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. (3.16) gives the instantaneous translational velocity of the endpoint of the position vector $\boldsymbol{p}$:

$$ {}_a\dot{\boldsymbol{p}} = {}^b_a\dot{\boldsymbol{R}}\, {}_b\boldsymbol{p} = {}^b_a\dot{\boldsymbol{R}}\ \left({}^a_b\boldsymbol{R}\, {}_a\boldsymbol{p}\right). \tag{3.17}$$

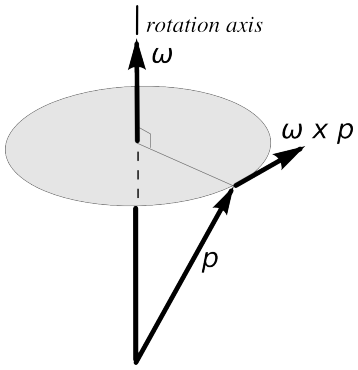

Figure 3.6: Translational velocity of a point fixed to a rigid body rotating with an angular velocity $\boldsymbol{\omega}$.

4. Alternatively, Fig. 3.6 shows that the same translational velocity is given by:

$$\begin{aligned} \dot{\boldsymbol{p}} &= \boldsymbol{\omega} \times \boldsymbol{p} \\ &= [\boldsymbol{\omega}]\, \boldsymbol{p}. \end{aligned} \tag{3.18}$$

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

$$[\boldsymbol{\omega}] = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}. \tag{3.19}$$

Hence, the following linear relationships result from Eqs (3.17) and (3.18):

$$[{}_a\boldsymbol{\omega}] = {}^b_a\dot{\boldsymbol{R}}\, {}^b_a\boldsymbol{R}^{-1}, \quad \text{or} \quad {}^b_a\dot{\boldsymbol{R}} = [{}_a\boldsymbol{\omega}]\, {}^b_a\boldsymbol{R}. \tag{3.20}$$

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*, [215]. 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.

It is no coincidence that $\boldsymbol{\omega}$ and $\boldsymbol{p}$ intersect in Figure 3.6: the intersection point does not move during the rotation, and all motions with one fixed point are pure rotations.

3.10 Exponential and logarithm

This Section gives a coordinate representation for the *exponential* and *logarithm* mappings in the case of an *angular velocity* about a frame axis, [228]. 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. (3.3),

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

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. (3.19): the solution of Eq. (3.20) is $\boldsymbol{R}(t) = C \exp([\boldsymbol{\omega}]t), C \in \mathbb{R}$. The angular motion starts with $\boldsymbol{R}(t=0)$ at time $t = 0$, such that $\exp([\boldsymbol{\omega}]t) = I_{3\times 3}$ and thus $C = 1$. Hence,

$$\boldsymbol{R}(X, \alpha_t) = \exp([\boldsymbol{\omega}]t). \tag{3.21}$$

An alternative, coordinate-based proof consists of filling in

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

in the Taylor series of the matrix exponential:

$$\exp(\boldsymbol{A}) = \mathbf{1} + \boldsymbol{A} + \frac{\boldsymbol{A}^2}{2!} + \frac{\boldsymbol{A}^3}{3!} + \ldots \tag{3.22}$$

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

$$\boldsymbol{A}^2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\omega_x^2 t^2 & 0 \\ 0 & 0 & -\omega_x^2 t^2 \end{pmatrix}, \quad \boldsymbol{A}^3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -\omega_x^3 t^3 \\ 0 & \omega_x^3 t^3 & 0 \end{pmatrix} = -\omega_x^2 t^2 \boldsymbol{A}. \tag{3.23}$$

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.

3.11 Infinitesimal rotation

Equations (3.21) and (3.22) 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. (3.22) after the linear term gives:

$$\boldsymbol{R}_\Delta = \begin{pmatrix} 1 & -\delta_z & \delta_y \\ \delta_z & 1 & \delta_x \\ -\delta_y & \delta_x & 1 \end{pmatrix}. \tag{3.24}$$

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

3.12 Euler angles

The previous Sections used the rotation matrix as a non-minimal 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.

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 *minimal representations.* Around 1750, Euler proved that the relative orientation of two coordinate systems can be specified by a set of *three* independent angles, [84, 203]. This results in the large set of three-angle orientation representations (all of them are called *Euler angles*!) discussed in this Section.

Euler's Theorem, 1775.
Any displacement of a rigid body that leaves one point fixed is a rotation about some axis, *[5, 24, 84].*

This theorem predicts the existence of a so-called *equivalent axis* and the corresponding *equivalent rotation angle*, discussed in more detail in Section 3.18. "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.

3.13 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 3.8 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. 3.7). 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 (3.3) and (3.5), 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. 3.7). The total resulting rotation matrix ${}^{ee}_{bs}\boldsymbol{R}$ is found from applying Eq. (3.12) 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. (3.1).

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 $\{a\}$.

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 ${}^{ee}_{bs}\boldsymbol{R}\ {}_{ee}\boldsymbol{x}^{ee} = {}^{ee}_{bs}\boldsymbol{R}\ (1\,0\,0)^T$.

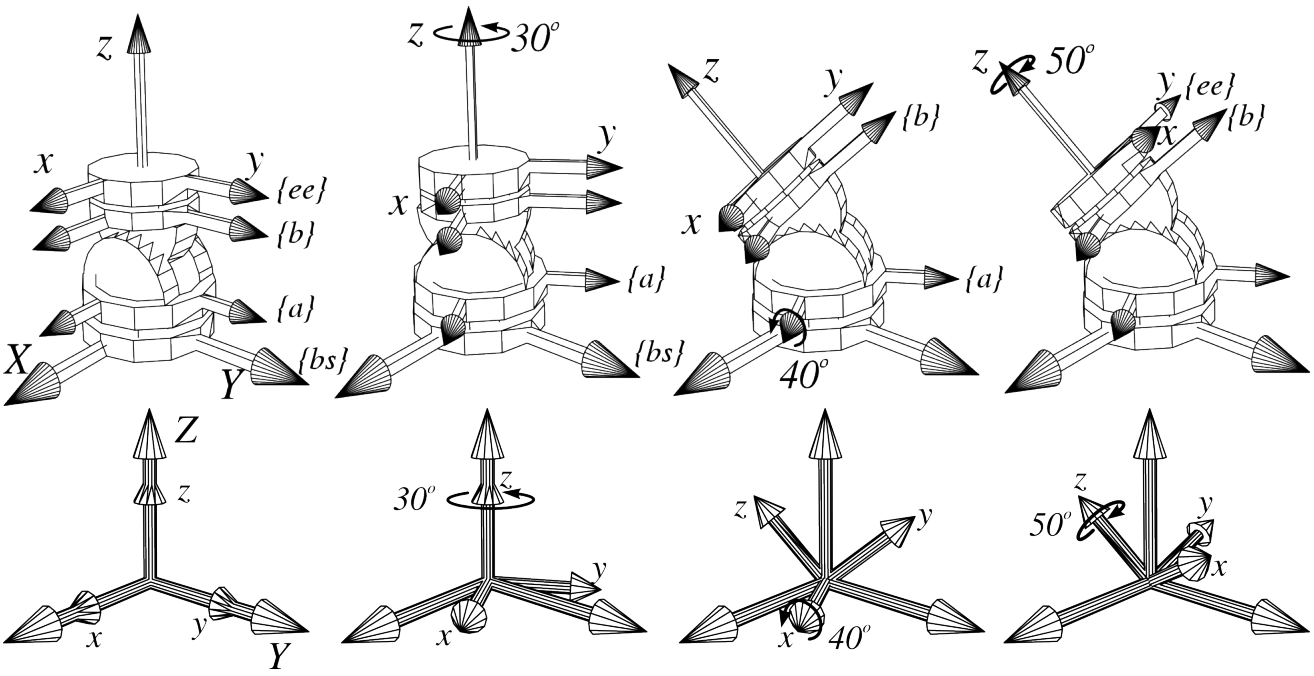

Figure 3.7: Two different views on a serial connection of three revolute joints with coinciding rotation axes, implementing the composition of three subsequent rotations, about the "moving" z, x and z axes. Such "ZXZ wrists" are often found in industrial robots,

4. Similarly for $\boldsymbol{y}^{ee}$ and $\boldsymbol{z}^{ee}$.

Hence

$$
\begin{aligned}
{}^{ee}_{bs}\boldsymbol{R} &= {}^{a}_{bs}\boldsymbol{R}\,{}^{b}_{a}\boldsymbol{R}\,{}^{ee}_{b}\boldsymbol{R} \\
&= \boldsymbol{R}(Z,\alpha)\,\boldsymbol{R}(X,\beta)\,\boldsymbol{R}(Z,\gamma) && (3.25)\\
&= \begin{pmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_\beta & -s_\beta \\ 0 & s_\beta & c_\beta \end{pmatrix} \begin{pmatrix} c_\gamma & -s_\gamma & 0 \\ s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} && (3.26)\\
&= \begin{pmatrix} c_\gamma c_\alpha - s_\gamma c_\beta s_\alpha & -s_\gamma c_\alpha - c_\gamma c_\beta s_\alpha & s_\beta s_\alpha \\ c_\gamma s_\alpha + s_\gamma c_\beta c_\alpha & -s_\gamma s_\alpha + c_\gamma c_\beta cP\alpha & -s_\beta c_\alpha \\ s_\gamma s_\beta & c_\gamma s_\beta & c_\beta \end{pmatrix}, && (3.27)
\end{aligned}
$$

with the obvious abbreviations $c_\alpha = \cos(\alpha)$, etc. Eq. (3.27) 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 3.7 to a desired Cartesian orientation, i.e., a desired orientation $\boldsymbol{R}$ is given, and the corresponding angles α, β and γ are sought. This inverse can be derived by inspection. α follows from the ratio of $\boldsymbol{R}_{13}(= \sin(\beta)\sin(\alpha))$ and $\boldsymbol{R}_{23}(= -\sin(\beta)\cos(\theta))$:

$$\alpha = \operatorname{atan2}(\boldsymbol{R}_{13}, -\boldsymbol{R}_{23}), \tag{3.28}$$

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. (3.27):

$$\beta = \operatorname{atan2}(-\boldsymbol{R}_{23}\cos(\alpha) + \boldsymbol{R}_{13}\sin(\alpha), \boldsymbol{R}_{33}). \tag{3.29}$$

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

$$\gamma = \operatorname{atan2}(\boldsymbol{R}_{31}, \boldsymbol{R}_{32}). \tag{3.30}$$

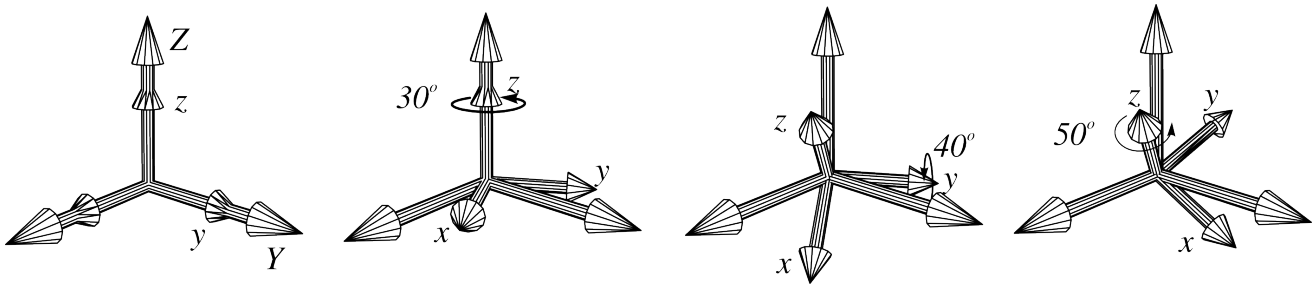

Figure 3.8: The ZYZ Euler angles.

Note the bad numerical conditioning for small β (i.e., a so-called *representation singularity*, or *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 (3.28)–(3.30) are not well defined.

Physically, this corresponds to situation in which the first and third axes of the wrist in Figure 3.7 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.

The representation singularity mentioned above is a *real* physical problem for robot wrists that have this mechanical construction. However, this "wrist" often occurs in purely mathematical representations of orientation as an intuitive virtual model, and then the physical problem does not manifest itself (there *is* no mechanics in this case!) but the mathematical problem remains. This problem could be avoided *locally*, by choosing another set of Euler angles; but any other set would have *its* singularities at some other configuration. Hence, singularity avoidance is possible, but requires attention from the programmer.

The inverse calculation in Eqs (3.28)–(3.30) also has a *second solution*, i.e., a different set of Euler angles that gives the same orientation: $\cos(\beta), \sin(\beta)\sin(\alpha)$ and $-\sin(\beta)\cos(\alpha)$ 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. 3.8 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, [203].)

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. 3.8):

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, [84].)
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.)

3.14 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:

$$\boldsymbol{R}(ZXZ,\alpha,\beta,\gamma) = \boldsymbol{Z}(Z,\gamma)\,\boldsymbol{R}(X,\beta)\,\boldsymbol{R}(Z,\alpha) \tag{3.31}$$

$$= \begin{pmatrix} c_\gamma & -s_\gamma & 0 \\ s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_\beta & -s_\beta \\ 0 & s_\beta & c_{beta} \end{pmatrix} \begin{pmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.32}$$

$$= \begin{pmatrix} c_\gamma c_\alpha - s_\gamma c_\beta s_\alpha & -c_\gamma s_\alpha - s_\gamma c_\beta c_\alpha & s_\gamma s_\beta \\ s_\gamma c_\alpha + c_\gamma c_\beta s_\alpha & -s_\gamma s_\alpha + c_\gamma c_\beta c_\alpha & -c_\gamma s_\beta \\ s_\beta s_\alpha & s_\beta c_\alpha & c_\beta \end{pmatrix}. \tag{3.33}$$

Note the difference with Eq. (3.27). 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 summary, Eqs (3.25) and (3.31) express Euler angle constructions with *moving and fixed axes:*

$$R^m_{zxz}(\alpha,\beta,\gamma) = \boldsymbol{R}(Z,\alpha)\boldsymbol{R}(X,\beta)\boldsymbol{R}(Z,\gamma), \tag{3.34}$$

and

$$R^f_{zxz}(\alpha,\beta,\gamma) = \boldsymbol{R}(Z,\gamma)\boldsymbol{R}(X,\beta)\boldsymbol{R}(Z,\alpha). \tag{3.35}$$

where superscripts "m" and "f" indicate that the rotations take place about moving, respectively fixed frame axes. The subscript denotes the order of the rotations, in the same order as the parameters of the operators which are the corresponding rotation angles.

Bryant angles. In some application domains, XYZ Euler angles are given the name of Bryant angles, [279, 187].

Roll-Pitch-Yaw angles (RPY). 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 (3.28)–(3.30). This situation is common in sea navigation, and hence, *Roll-Pitch-Yaw* angles have been introduced many centuries ago already, to describe rotations about three *orthogonal* axes *fixed* to the moving object or vehicle. The name "roll, pitch, yaw" still reflects its nautical origin: the 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. 3.9). 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. 3.10). Hence, the rotation matrix corresponding to the ZYX or Roll-Pitch-Yaw (Euler) angles[1] is

$$\begin{aligned} \boldsymbol{R}(RPY,r,p,y) &= \boldsymbol{R}(ZYX,y,p,r) \\ &= \boldsymbol{R}(Z,y)\,\boldsymbol{R}(Y,p)\,\boldsymbol{R}(X,r) \\ &= \begin{pmatrix} c_y & -s_y & 0 \\ s_y & c_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_p & 0 & s_p \\ 0 & 1 & 0 \\ -s_p & 0 & c_p \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_r & -s_r \\ 0 & s_r & c_r \end{pmatrix} \\ &= \begin{pmatrix} c_y c_p & c_y s_p s_r - s_y c_r & c_y s_p c_r + s_y s_r \\ s_y c_p & s_y s_p s_r + c_y c_r & s_y s_p c_r - c_y s_r \\ -s_p & c_p s_r & c_p c_r \end{pmatrix}. \end{aligned} \tag{3.36}$$

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. (3.36); for example,

$$r = \text{atan2}(\boldsymbol{R}_{32}, \boldsymbol{R}_{33}), \tag{3.37}$$

$$y = \text{atan2}(\boldsymbol{R}_{21}, \boldsymbol{R}_{11}), \tag{3.38}$$

$$p = \text{atan2}(-\boldsymbol{R}_{31}, c_y\boldsymbol{R}_{11} + s_y\boldsymbol{R}_{21}). \tag{3.39}$$

[1] Some publications indeed speak about RPY *Euler* angles.

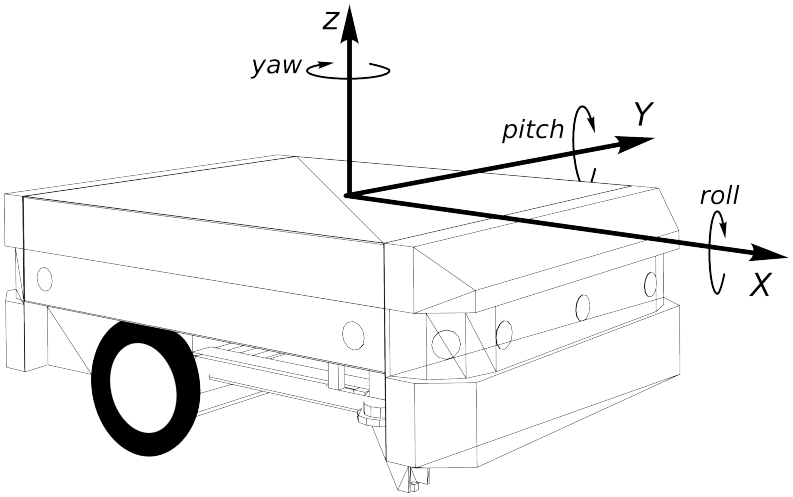

Figure 3.9: Roll-Pitch-Yaw angles for a mobile robot.

Note some similarities with the Euler angles of Eqs (3.28)–(3.30):

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 $y + \pi$.

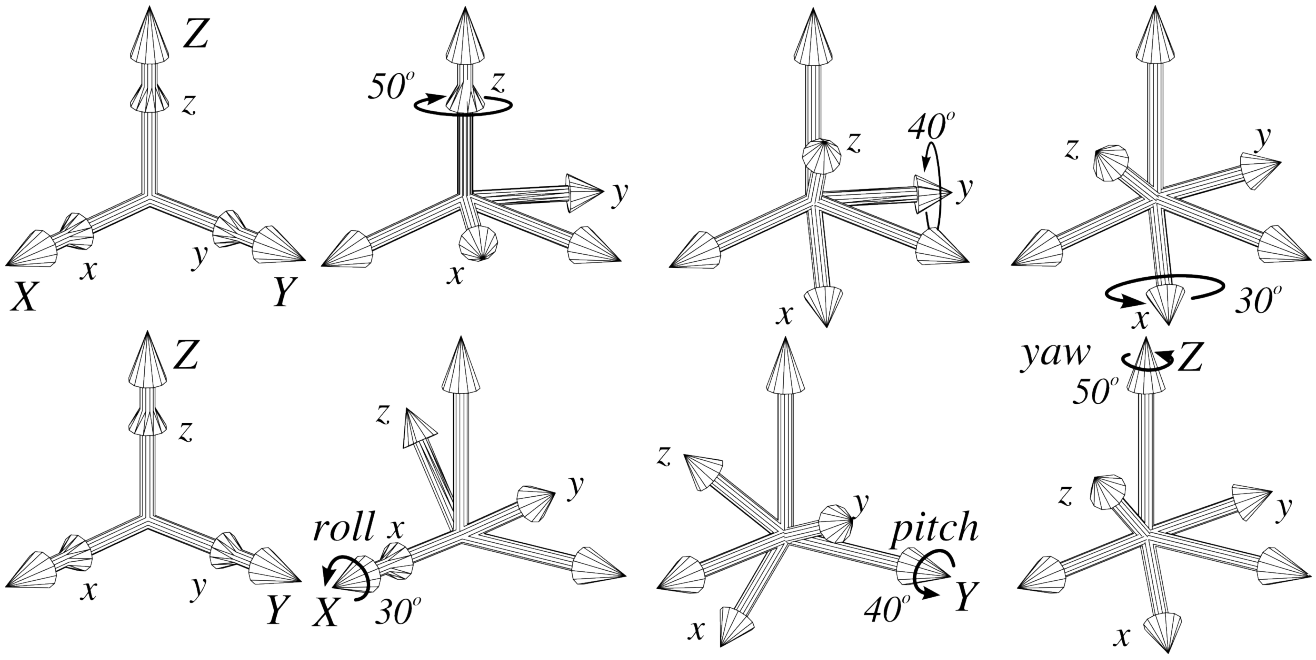

Figure 3.10: *ZYX* Euler angles (top) and Roll-Pitch-Yaw angles (bottom), both corresponding to the same orientation.

3.15 Properties of three-angle orientation representations

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 3.7 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.

The disadvantages are:

1. All Euler angle representations have *mathematical singularity* problems: no set of three angles can *globally* represent all orientations without singularity, [107]. 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. (3.36) is a continuous function of its Euler angle parameters.) For example, the robot wrist in Figure 3.7 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 (3.38)-(3.39) 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. This phenomenon was first experienced in the mechanisms ("gimbals") that were developed to support the compasses on ships, and there it got its name of *gimbal lock*.

2. 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

3. Euler angles are *not a vector*:

   - *Adding* two sets of Euler angles does not give the set of Euler angles that corresponds to the composed orientation. Obviously, also subtraction of Euler angle sets is not physically meaningful.
   - The *order* of rotations matters: composition of rotations is not commutative, while vector addition is.

   So, one should not blindly apply linear vector space properties to Euler angles!

3.16 Euler angle time rates and angular velocity

Equation (3.20) 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 Fig. 3.7. 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 ${}^a_{bs}\boldsymbol{R}\ (1\ 0\ 0)^T = (\cos(\alpha)\ \ \sin(\alpha)\ \ 0)^T$. 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 ${}^a_{bs}\boldsymbol{R}\ {}^b_a\boldsymbol{R}\,(0\ 0\ 1)^T = (\sin(\beta)\sin(\alpha)\ \ -\sin(\beta)\cos(\alpha)\ \ \cos(\beta))^T$. Summing these three angular velocity contributions, the total angular velocity three-vector $\boldsymbol{\omega}$ is

$$\begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \begin{pmatrix} 0 & \cos(\alpha) & \sin(\beta)\sin(\alpha) \\ 0 & \sin(\alpha) & -\sin(\beta)\cos(\alpha) \\ 1 & 0 & \cos(\beta) \end{pmatrix} \begin{pmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{pmatrix}. \tag{3.40}$$

3.16.1 Inverse relationship

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

$$\begin{pmatrix}\dot{\alpha}\\ \dot{\beta}\\ \dot{\gamma}\end{pmatrix} = \begin{pmatrix} -\dfrac{\sin(\alpha)\cos(\beta)}{\sin(\beta)} & \dfrac{\cos(\alpha)\cos(\beta)}{\sin(\beta)} & 1\\ \cos(\alpha) & \sin(\alpha) & 0\\ \dfrac{\sin(\alpha)}{\sin(\beta)} & -\dfrac{\cos(\alpha)}{\sin(\beta)} & 0\end{pmatrix}\begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix}. \tag{3.41}$$

Similarly to the before-mentioned singularity analysis, this inverse becomes *singular* for $\beta = 0$, i.e., when the first and third joint axes of the spherical wrist in Fig. 3.7 line up. In this configuration, no angular velocity about the Y axis is possible.

3.16.2 Angular velocity is a vector

In contrast to the Euler angle sets, the angular velocity *is* a vector (which we denote by $\boldsymbol{\omega}$): it makes physical sense to add two angular velocities together into a third angular velocity which corresponds to the composition of both individual angular velocities.

3.17 Integrability of angular velocity

Integrating the angular velocity over a certain amount of time results in a change of Euler angles. But: the angular velocity three-vector $\boldsymbol{\omega}$ is *not exact* (which is a term from *integration theory*, i.e., it is not the time derivative of any Euler angle set, [104, p. 347].

The non-exactness is proved as follows. Consider, for example, the Y component of $\boldsymbol{\omega}$ in Eq. (3.40). In "differential" form, this gives

$$\sin(\alpha)d\beta - \sin(\beta)\cos(\alpha)d\gamma = A(\alpha,\beta,\gamma)d\alpha + B(\alpha,\beta,\gamma)d\beta + C(\alpha,\beta,\gamma)d\gamma. \tag{3.42}$$

Such a differential form is integrable (or *exact* [33, 215, 222, 248]) if and only if

$$\frac{\partial A}{\partial \beta} = \frac{\partial B}{\partial \alpha}, \tag{3.43}$$

as well as all similar combinations with A, B and C. That this condition is not always satisfied is easily checked from Eq. (3.42), for example:

$$\frac{\partial C}{\partial \beta} = -\cos(\beta)\cos(\alpha), \quad \text{but} \quad \frac{\partial B}{\partial \gamma} = 0. \tag{3.44}$$

However, *integrating factors* such as in Eq. (3.40) exist, so that there *is* a mathematical way to derive a change in orientation by applying a certain angular velocity during a certain time. (Which corresponds to our physical intuition, of course.)

3.18 Equivalent axis and equivalent angle of rotation

Euler's Theorem (Sect. 3.12) 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?

3.18.1 Forward relation

If the axis $\boldsymbol{e}^{eq} = (\boldsymbol{e}_x^{eq}\ \boldsymbol{e}_y^{eq}\ \boldsymbol{e}_z^{eq})^T$ 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. 3.11):

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 $\alpha = -\arctan(\boldsymbol{e}_y^{eq}/\boldsymbol{e}_x^{eq})$.

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 $\beta = \arctan\left(\boldsymbol{e}_z^{eq}/\left((\boldsymbol{e}_x^{eq})^2 + (\boldsymbol{e}_y^{eq})^2\right)\right)$.

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

$$\boldsymbol{R}(\boldsymbol{e}^{eq},\theta) = \boldsymbol{R}(Z,-\alpha)\boldsymbol{R}(Y,-\beta)\boldsymbol{R}(X,\theta)\boldsymbol{R}(Y,\beta)\boldsymbol{R}(Z,\alpha), \tag{3.45}$$

or

$$\boldsymbol{R}(\boldsymbol{e}^{eq},\theta) = \begin{pmatrix} (\boldsymbol{e}_x^{eq})^2 v_t + c_t & \boldsymbol{e}_x^{eq}\boldsymbol{e}_y^{eq} v_t - \boldsymbol{e}_z^{eq} s_t & \boldsymbol{e}_x^{eq}\boldsymbol{e}_z^{eq} v_t + \boldsymbol{e}_y^{eq} s_t \\ \boldsymbol{e}_x^{eq}\boldsymbol{e}_y^{eq} v_t + \boldsymbol{e}_z^{eq} s_t & (\boldsymbol{e}_y^{eq})^2 v_t + c_t & \boldsymbol{e}_y^{eq}\boldsymbol{e}_z^{eq} v_t - \boldsymbol{e}_x^{eq} s_t \\ \boldsymbol{e}_x^{eq}\boldsymbol{e}_z^{eq} v_t - \boldsymbol{e}_y^{eq} s_t & \boldsymbol{e}_y^{eq}\boldsymbol{e}_z^{eq} v_t + \boldsymbol{e}_x^{eq} s_t & (\boldsymbol{e}_z^{eq})^2 v_t + c_t \end{pmatrix}. \tag{3.46}$$

c_t and s_t are shorthand notations for $\cos(\theta)$ and $\sin(\theta)$, respectively. v_t is the "verse of theta," which is equal to $1 - c_t$.

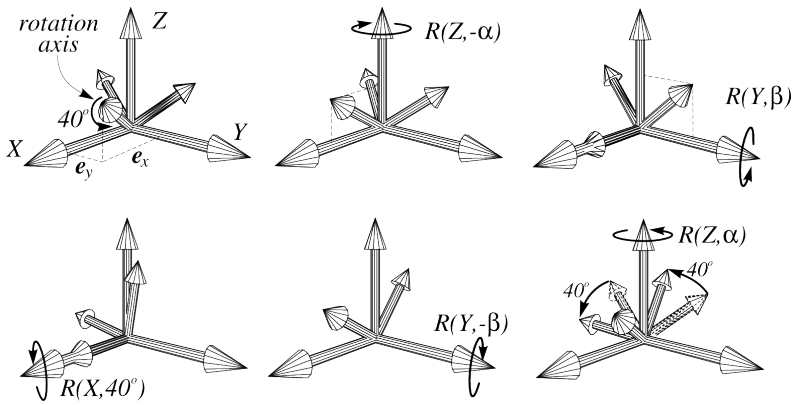

Figure 3.11: Rotation about an arbitrary axis is equivalent to a sequence of five rotations about the fixed axes.

3.18.2 Inverse relation

The transformations from rotation matrix to equivalent axis parameters are deduced from Eq. (3.46), via the following observations, [53, 84, 104, 173]:

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

$$\text{trace}\left(\boldsymbol{R}(\boldsymbol{e}^{eq},\theta)\right) = 1 + 2cos(\theta). \tag{3.47}$$

Hence

$$\theta = \arccos\left(\frac{\text{trace}\left(\boldsymbol{R}(\boldsymbol{e}^{eq},\theta)\right) - 1}{2}\right). \tag{3.48}$$

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

$$\begin{array}{rcl} \boldsymbol{R}_{32}(\boldsymbol{e}^{eq},\theta) - \boldsymbol{R}_{23}(\boldsymbol{e}^{eq},\theta) &=& 2\boldsymbol{e}_x^{eq}\sin(\theta), \\ \boldsymbol{R}_{13}(\boldsymbol{e}^{eq},\theta) - \boldsymbol{R}_{31}(\boldsymbol{e}^{eq},\theta) &=& 2\boldsymbol{e}_y^{eq}\sin(\theta), \\ \boldsymbol{R}_{21}(\boldsymbol{e}^{eq},\theta) - \boldsymbol{R}_{12}(\boldsymbol{e}^{eq},\theta) &=& 2\boldsymbol{e}_z^{eq}\sin(\theta). \end{array} \tag{3.49}$$

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

$$\boldsymbol{e}^{eq} = \frac{1}{2\sin(\theta)} \begin{pmatrix} \boldsymbol{R}_{32}(\boldsymbol{e}^{eq}, \theta) - \boldsymbol{R}_{23}(\boldsymbol{e}^{eq}, \theta) \\ \boldsymbol{R}_{13}(\boldsymbol{e}^{eq}, \theta) - \boldsymbol{R}_{31}(\boldsymbol{e}^{eq}, \theta) \\ \boldsymbol{R}_{21}(\boldsymbol{e}^{eq}, \theta) - \boldsymbol{R}_{12}(\boldsymbol{e}^{eq}, \theta) \end{pmatrix}. \tag{3.50}$$

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.

3.19 Time derivative

Equation (3.45) 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. (3.20) 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:

$$\begin{aligned} [\boldsymbol{\omega}^{eq}] &= {}^R\dot{\boldsymbol{R}}(\boldsymbol{e}^{eq}, \theta)\ \boldsymbol{R}^{-1}(\boldsymbol{e}^{eq}, \theta) \\ &= \Big\{\boldsymbol{R}(Z, -\alpha)\boldsymbol{R}(Y, -\beta)\dot{\boldsymbol{R}}(X, \theta)\boldsymbol{R}(Y, \beta)\boldsymbol{R}(Z, \alpha)\Big\}\ \{\boldsymbol{R}(Z, -\alpha)\boldsymbol{R}(Y, -\beta)\boldsymbol{R}(X, \theta)\boldsymbol{R}(Y, \beta)\boldsymbol{R}(Z, \alpha)\} \\ &= \boldsymbol{R}(Z, -\alpha)\boldsymbol{R}(Y, -\beta)\,[\boldsymbol{\omega}_x]\,\boldsymbol{R}(Y, \beta)\boldsymbol{R}(Z, \alpha). \end{aligned} \tag{3.51}$$

3.20 Similarity transformations

The procedure applied in deriving Eq. (3.45) works for general transformations too, not just rotations about frame axes. So, the following Chapters of this book will often use these *similarity transformations*: often, a general transformation T can be written as

$$T = S^{-1}T'S, \tag{3.52}$$

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

3.21 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 3.10 proved that any rotation about one of the *frame* axes corresponds to the exponentiation of an angular velocity about this axis, Eq. (3.21). According to Eq. (3.45), 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. (3.52), 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 (3.22) and (3.51) then imply that

$$\begin{aligned} \boldsymbol{R}(\boldsymbol{e}^{eq}, \theta) &= \boldsymbol{S}^{-1}\exp(\boldsymbol{A})\boldsymbol{S} \\ &= \boldsymbol{S}^{-1}\left(\mathbf{1} + \boldsymbol{A} + \frac{\boldsymbol{A}^2}{2!} + \ldots\right)\boldsymbol{S} \\ &= \mathbf{1} + \left(\boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{S}\right) + \frac{\boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{S}\boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{S}}{2!} + \ldots \\ &= \exp\left(\boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{S}\right) \\ &= \exp\left([\boldsymbol{\omega}^{eq}]\right). \end{aligned} \tag{3.53}$$

$\boldsymbol{\omega}^{eq}$ 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.

3.22 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, [173, 193]. (Hence, it is a *structural* (or "physical") 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, 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 $\boldsymbol{R} = \left(\boldsymbol{R}^1\right)^{-1} \boldsymbol{R}^2$. In other words, this distance is the magnitude of the angular velocity that can close the orientation gap in one unit of time.

It can be proven 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.

3.23 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.

3.23.1 Definition and use

This interesting orientation representation is the *four*-parameter set of *unit quaternions*, also called *Euler-(Rodrigues) parameters*, [45, 241]. 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:

$$\boldsymbol{q}(\boldsymbol{e}^{eq}, \theta) = \begin{pmatrix} \boldsymbol{q}_v \\ q \end{pmatrix} = \begin{pmatrix} \sin(\frac{\theta}{2})\ \boldsymbol{e}_x^{eq} \\ \sin(\frac{\theta}{2})\ \boldsymbol{e}_y^{eq} \\ \sin(\frac{\theta}{2})\ \boldsymbol{e}_z^{eq} \\ \cos(\frac{\theta}{2}) \end{pmatrix}. \tag{3.54}$$

Since, in general, two equivalent rotation angles and axes exist for every rotation (Sect. 3.18.2), 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:

$$\boldsymbol{q}^T \boldsymbol{q} = \boldsymbol{q}_v^T \boldsymbol{q}_v + q^2 = 1. \tag{3.55}$$

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 [100] even *before* the dot product and cross product between three-vectors were introduced by Josiah Willard Gibbs (1839–1903), [57]. The parameterization of rotations by means of quaternions was already described by Olinde Rodrigues in 1840 [89, 218, 254], and even earlier by Johann Carl Friedrich Gauss (1777–1855), who didn't bother to publish about them.[2] 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 $i = \sqrt{-1}$, the quaternions have four generating four-vectors $\mathbf{1}, \boldsymbol{i}, \boldsymbol{j}$ and $\boldsymbol{k}$:

$$\boldsymbol{i} = \boldsymbol{q}(\boldsymbol{e}_x, \pi) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \boldsymbol{j} = \boldsymbol{q}(\boldsymbol{e}_y, \pi) = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \boldsymbol{k} = \boldsymbol{q}(\boldsymbol{e}_z, \pi) = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \mathbf{1} = \boldsymbol{q}(\times, 0) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{3.56}$$

[2] Reference needed!

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

$$\boldsymbol{ij} = \boldsymbol{k}, \quad \boldsymbol{ij} = -\boldsymbol{ji}, \quad \boldsymbol{ii} = -1, \tag{3.57}$$

plus all cyclic permutations.

3.23.2 Why quaternions?

The most prominent reason to use quaternions in robotics (and other domains, such as computer animation, or computer vision) is that they give a *singularity free* mathematical representation. In addition, a unit quaternion is the orientation representation with the *smallest* number of parameters that can represent all orientations *without singularity*, [5, 249]. 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. (3.54) 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. 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. (3.50).

3.23.3 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. (3.57):

$$\boldsymbol{q}^1\boldsymbol{q}^2 = \begin{pmatrix} y^1z^2 - y^2z^1 + x^1s^2 + x^2s^1 \\ z^1x^2 - z^2x^1 + y^1s^2 + y^2s^1 \\ x^1y^2 - x^2y^1 + z^1s^2 + z^2s^1 \\ s^1s^2 - x^1x^2 - y^1y^2 - z^1z^2 \end{pmatrix}. \tag{3.58}$$

(See e.g., [84, 100, 104, 161, 187, 188] for more detailed algebraic discussions.)[3] The right-hand side can be re-organised in a scalar part and a vector part:

$$\boldsymbol{q}^1\boldsymbol{q}^2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ s^1s^2 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 0 \\ x^1x^2 + y^1y^2 + z^1z^2 \end{pmatrix} + s^1\begin{pmatrix} x^2 \\ y^2 \\ z^2 \\ 0 \end{pmatrix} + s^2\begin{pmatrix} x^1 \\ y^1 \\ z^1 \\ 0 \end{pmatrix} + \begin{pmatrix} y^1z^2 - y^2z^1 \\ z^1x^2 - z^2x^1 \\ x^1y^2 - x^2y^1 \\ 0 \end{pmatrix} \tag{3.59}$$

$$= \left(q^1q^2 - \boldsymbol{q}_v^1 \cdot \boldsymbol{q}_v^2\right) + \left(q^1\boldsymbol{q}_v^2 + q^2\boldsymbol{q}_v^1 + \boldsymbol{q}_v^1 \times \boldsymbol{q}_v^2\right). \tag{3.60}$$

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*, [124, 161] (which is alway handy to *compose* operations):

$$\boldsymbol{q}^1\boldsymbol{q}^2 = \boldsymbol{Q}_l^1\boldsymbol{q}^2 = \boldsymbol{Q}_r^2\boldsymbol{q}^1, \tag{3.61}$$

[3] These details should be added in an Appendix!

with

$$\boldsymbol{Q}_l^1 = \begin{pmatrix} s^1 & -z^1 & y^1 & x^1 \\ z^1 & s^1 & -x^1 & y^1 \\ -y^1 & x^1 & s^1 & z^1 \\ -x^1 & -y^1 & -z^1 & s^1 \end{pmatrix}, \qquad \boldsymbol{Q}_r^2 = \begin{pmatrix} s^2 & z^2 & -y^2 & x^2 \\ -z^2 & s^2 & x^2 & y^2 \\ y^2 & -x^2 & s^2 & z^2 \\ -x^2 & -y^2 & -z^2 & s^2 \end{pmatrix}. \tag{3.62}$$

$\boldsymbol{Q}^i$ 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

$$\boldsymbol{Q}(\boldsymbol{q}^{-1}) = \boldsymbol{Q}^T(\boldsymbol{q}), \quad \boldsymbol{Q}_r^{-1} = \boldsymbol{Q}_r^T, \quad \boldsymbol{Q}_l^{-1} = \boldsymbol{Q}_l^T. \tag{3.63}$$

3.23.4 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, [104, 161],

$$\boldsymbol{p\prime} = \boldsymbol{q}\boldsymbol{p}\boldsymbol{q}^{-1}, \qquad \boldsymbol{p\prime} = \boldsymbol{Q}_l \boldsymbol{Q}_r^T \boldsymbol{p}. \tag{3.64}$$

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

$$\boldsymbol{Q}_l = \begin{pmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2})\,\boldsymbol{e}_z & \sin(\frac{\theta}{2})\,\boldsymbol{e}_y & \sin(\frac{\theta}{2})\,\boldsymbol{e}_x \\ \sin(\frac{\theta}{2})\,\boldsymbol{e}_z & \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2})\,\boldsymbol{e}_x & \sin(\frac{\theta}{2})\,\boldsymbol{e}_y \\ -\sin(\frac{\theta}{2})\,\boldsymbol{e}_y & \sin(\frac{\theta}{2})\,\boldsymbol{e}_x & \cos(\frac{\theta}{2}) & \sin(\frac{\theta}{2})\,\boldsymbol{e}_z \\ -\sin(\frac{\theta}{2})\,\boldsymbol{e}_x & -\sin(\frac{\theta}{2})\,\boldsymbol{e}_y & -\sin(\frac{\theta}{2})\,\boldsymbol{e}_z & \cos(\frac{\theta}{2}) \end{pmatrix}, \tag{3.65}$$

$$\text{and} \quad \boldsymbol{Q}_r = \begin{pmatrix} \cos(\frac{\theta}{2}) & \sin(\frac{\theta}{2})\,\boldsymbol{e}_z & -\sin(\frac{\theta}{2})\,\boldsymbol{e}_y & \sin(\frac{\theta}{2})\,\boldsymbol{e}_x \\ -\sin(\frac{\theta}{2})\,\boldsymbol{e}_z & \cos(\frac{\theta}{2}) & \sin(\frac{\theta}{2})\,\boldsymbol{e}_x & \sin(\frac{\theta}{2})\,\boldsymbol{e}_y \\ \sin(\frac{\theta}{2})\,\boldsymbol{e}_y & -\sin(\frac{\theta}{2})\,\boldsymbol{e}_x & \cos(\frac{\theta}{2}) & \sin(\frac{\theta}{2})\,\boldsymbol{e}_z \\ -\sin(\frac{\theta}{2})\,\boldsymbol{e}_x & -\sin(\frac{\theta}{2})\,\boldsymbol{e}_y & -\sin(\frac{\theta}{2})\,\boldsymbol{e}_z & \cos(\frac{\theta}{2}) \end{pmatrix}. \tag{3.66}$$

Hence, Eq. (3.64) gives

$$\boldsymbol{p\prime} = \begin{pmatrix} \cos^2(\frac{\theta}{2}) - \sin^2(\frac{\theta}{2})(\boldsymbol{e}_z^2 + \boldsymbol{e}_y^2 - \boldsymbol{e}_x^2) & -2(\cos(\frac{\theta}{2})\sin(\frac{\theta}{2})\boldsymbol{e}_z + \sin^2(\frac{\theta}{2})\boldsymbol{e}_x\boldsymbol{e}_y) & 2(\cos(\frac{\theta}{2})\sin(\frac{\theta}{2})\boldsymbol{e}_y + \sin^2(\frac{\theta}{2})\boldsymbol{e}_x\boldsymbol{e}_z) & 0 \\ 2(\cos(\frac{\theta}{2})\sin(\frac{\theta}{2})\boldsymbol{e}_z + \sin^2(\frac{\theta}{2})\boldsymbol{e}_x\boldsymbol{e}_y) & \cos^2(\frac{\theta}{2}) - \sin^2(\frac{\theta}{2})(\boldsymbol{e}_z^2 + \boldsymbol{e}_x^2 - \boldsymbol{e}_y^2) & -2(\cos(\frac{\theta}{2})\sin(\frac{\theta}{2})\boldsymbol{e}_x - \sin^2(\frac{\theta}{2})\boldsymbol{e}_y\boldsymbol{e}_z) & 0 \\ -2(\cos(\frac{\theta}{2})\sin(\frac{\theta}{2})\boldsymbol{e}_y - \sin^2(\theta)\boldsymbol{e}_x\boldsymbol{e}_z) & 2(\cos(\theta)\sin(\theta)\boldsymbol{e}_x + \sin^2(\theta)\boldsymbol{e}_y\boldsymbol{e}_z) & \cos^2(\frac{\theta}{2}) - \sin^2(\frac{\theta}{2})(\boldsymbol{e}_y^2 + \boldsymbol{e}_x^2 - \boldsymbol{e}_z^2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \boldsymbol{p}. \tag{3.67}$$

With the obvious substitutions $\cos^2(\frac{\theta}{2}) - \sin^2(\frac{\theta}{2}) = \cos(\theta)$, and $2\sin^2(\frac{\theta}{2}) = 1 - \cos(\theta) = v_\theta$, this result is equivalent to Eq. (3.46), 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., $\boldsymbol{q}^1(\boldsymbol{q}^2\boldsymbol{q}^3) = (\boldsymbol{q}^1\boldsymbol{q}^2)\boldsymbol{q}^3$.[4]

3.23.5 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 (3.18) and (3.65):

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{\Omega}_l \; \boldsymbol{q}, \tag{3.68}$$

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

[4] This calculation should be added in an Appendix!

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. (3.61), and
2. $\boldsymbol{Q}_r^{-1} = \boldsymbol{Q}_r^T$, Eq. (3.63).

Hence

$$\begin{pmatrix} \boldsymbol{\omega} \\ 0 \end{pmatrix} = 2\boldsymbol{Q}_r^T \dot{\boldsymbol{q}}. \tag{3.69}$$

$2\boldsymbol{Q}_r^T$ is the integrating factor. Note that the forward and inverse relationships (3.68) and (3.69) *never* become singular, while this is not the case for the Euler angle sets, see, for example, Eq. (3.41).

Chapter 4

Rigid body kinematics and dynamics

This Chapter presents the kinematics and dynamics of one single rigid body, since this provides the basic building blocks for the kinematics and dynamics of robots, and of kinematic chains in general, Chap. 5. A rigid body in the three-dimensional Euclidean space E^3 has six degrees-of-freedom *to represent* its *pose*: three in translation and three in rotation. But the same *physical* pose can be represented by an infinite number of different *mathematical coordinate choices*. Chapter 3 has discussed the three orientation degrees-of-freedom separately; this Chapter integrates these orientation degrees-of-freedom with the three translation degrees-of-freedom, following much the same route as in Chapter 3.

From the infinite amount of mathematical coordinate representations, many with only the minimum number of six parameters have been developed over the years. As with orientations, the fundamental physical properties of rigid body *pose* cannot be represented by linear six-dimensional vector spaces, i.e., with addition and/or multiplication of six-dimensional "vectors" as the standard operators in these spaces. However, the *velocity* of a rigid body *can* be represented by a six-dimensional vector (being a member of a linear vector space). The literature sometimes calls these six-dimensional rigid body velocity representations *screws*, *twists*, or, more often, One can say that a coordinate representation faithfully represents *the* pose, velocity or acceleration of a *rigid body*, if it contains all information that one needs to find the velocity and acceleration of *any point* moving together with the body. This Chapter explains what the *minimum* information is for different coordinate representations. For various reasons, one often also uses a *non-minimal matrix* representation to represent the properties of rigid body motion. The same trade-offs exist as in the orientation Chapter: improved properties on the one hand, but extra cost on the other hand, because of the need to carry along a larger amount of numerical values, and a larger number of constraints between these values.

4.1 Position and orientation—Pose

Orientations and their representations are not very intuitive in more than one respect, as explained in Chap. 3. However, extending the description to include translations turns out to require only a minor extra effort with respect to what a user needs to master to understand and to represent orientations.

4.1.1 Definition and use

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)* ${}^b_a\boldsymbol{R}$ (or *homogeneous transformation matrix*, or *homogeneous transform*, for short):

$$ {}^b_a\boldsymbol{T} = \begin{pmatrix} {}^b_a\boldsymbol{R} & {}_a\boldsymbol{p}^{a,b} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix}. \tag{4.1} $$

${}^b_a\boldsymbol{T}$ is the coordinate representation of a "*point*" in the six-dimensional space of all positions and orientations of a rigid body, 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

$$\begin{pmatrix} {}_a\boldsymbol{p} \\ 1 \end{pmatrix} = {}^b_a\boldsymbol{T} \begin{pmatrix} {}_b\boldsymbol{p} \\ 1 \end{pmatrix}. \tag{4.2}$$

This is obvious from Fig. 4.1. Note that Eq. (4.2) 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.

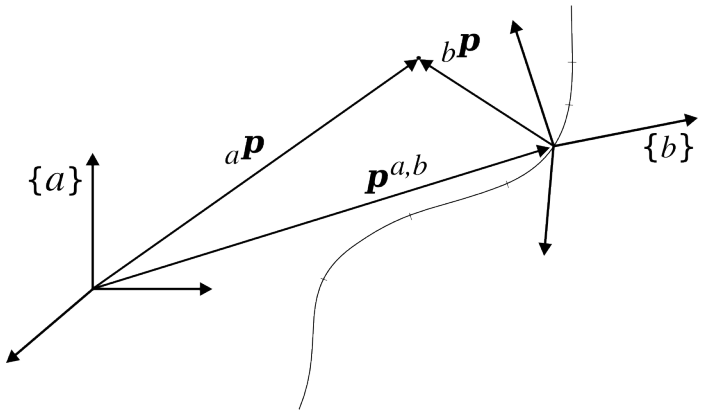

Figure 4.1: Frame $\{b\}$ moves with respect to frame $\{a\}$. The point $\boldsymbol{p}$ moves together with$\{b\}$.

4.1.2 Active and passive interpretation

As for screws (Sect. 2.7.7) and for rotations and orientations (Sect. 3.5), one can interpret a homogeneous transformation matrix both actively and passively. The passive interpretation is often connected to the terminology "pose," while the terminology "displacement" is more suggestive of an active interpretation.

4.1.3 Uniqueness

From the definition of the homogeneous transformation matrix, Eq. (4.1), and the uniqueness property of the rotation matrix, it follows that 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.

4.1.4 Inverse

Given the simple formula for the inverse of a rotation matrix, Eq. (3.7), it is straightforward to check that the *inverse* of a homogeneous transformation matrix ${}^b_a\boldsymbol{T}$ is

$${}^b_a\boldsymbol{T}^{-1} = \begin{pmatrix} {}^b_a\boldsymbol{R} & {}_a\boldsymbol{p}^{a,b} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix}^{-1} = \begin{pmatrix} {}^b_a\boldsymbol{R}^T & -{}^b_a\boldsymbol{R}^T{}_a\boldsymbol{p}^{a,b} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix}. \tag{4.3}$$

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

4.1.5 Non-minimal representation

A direct consequence of the non-minimality of the rotation matrix is that a homogeneous transformation matrix is not a minimal representation of a pose. The big advantage of using this non-minimal representation is that it has *no coordinate singularities*.

4.1.6 Composition of rigid body poses

The same reasoning as in Sect. 3.8 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

$$ {}^{tl}_{bs}\boldsymbol{T} = {}^{ee}_{bs}\boldsymbol{T}\,{}^{tl}_{ee}\boldsymbol{T}. \tag{4.4} $$

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. (4.2) and illustrated in Fig. 4.1. As with orientations, an important insight to get from this composition formula is that composition of poses is a *multiplicative operation*, and not an additive operation; this insight is often not made explicitly by most scientists and engineers, whose education has almost always been dominated (or even monopolized) by linear, additive system operations.

4.1.7 Finite displacement twist

Equation (4.1) uses a 4×4 matrix to represent a pose, thus it is a non-minimal coordinate choice. Other, minimal, orientation representations could be used instead of the rotation matrix, in order to give a minimal coordinate representation for a pose too. One frequently encountered minimal coordinate representation alternative for the non-minimal homogeneous transformation matrix is the *finite displacement*, also called "finite displacement *twist*". It is represented by the following six-dimensional coordinate "vector":

$$ \mathbf{t}_d = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ x \\ y \\ z \end{pmatrix}, \tag{4.5} $$

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. The finite displacement is *not* a member of a vector space, Sect. 3.15, since the first three components are an Euler angle set; hence, the pose corresponding to adding the coordinates of two finite displacement twists is not the same as the composition of both poses (or any other physical operation on the two poses for that matter).

4.1.8 Infinitesimal displacement twist

If one brings the final position and orientation of the moving body of the previous Section closer and closer to the initial position (or, equivalently, one considers the position and orientation of the body at two close instants in time) the screw axis is called the *instantaneous* screw axis (ISA) or *twist axis* of this infinitesimal displacement. The corresponding *infinitesimal displacement (twist)* is a six-dimensional coordinate "vector" denoted by $\mathbf{t}_\Delta$, and the *infinitesimal transformation matrix*, a 4×4 coordinate matrix denoted by $\boldsymbol{T}_\Delta$. They both describe small differences in *pose*:

$$ \mathbf{t}_\Delta = \begin{pmatrix} \delta_x \\ \delta_y \\ \delta_z \\ d_x \\ d_y \\ d_z \end{pmatrix}, \qquad \boldsymbol{T}_\Delta = \begin{pmatrix} 1 & -\delta_z & \delta_y & d_x \\ \delta_z & 1 & -\delta_x & d_y \\ -\delta_y & \delta_x & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.6} $$

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.

In contrast with the finite displacement twist of Sect. 4.1.7, the *infinitesimal* displacement twist *is* a vector, i.e., the operation of *addition* makes physical sense.

4.1.9 Non-commutation of translation and rotation

Translational and rotational displacements of a rigid body do not commute in general. However, the two motions described by the individual translational and rotational motion vectors do commute *if* both have the same

direction: first translating along the line of this direction and then rotating about it results in the same motion as performing the translation and rotation in the opposite order. So, when expressing a finite motion on the screw axis, this condition holds. However, this is an example of a "property" of one particular *representation*, that is not a *physical* (or "*invariant*") property: representing the same motion in another frame makes the "property" disappear, since the rotation and translation vectors change under changes of coordinate representations.

4.2 Velocity—Twist

The previous sections did not take *time* into account, only the geometrical displacements between rigid body poses. Figure 4.1 sketches a moving rigid body (or rather the motion of the *reference frame* $\{b\}$ attached to this body), with the small "ticks" on the trajectory indicating equal intervals in time. The rigid body traces a (timed) curve in the six-dimensional pose space, so this frame $\{b\}$ traces a curve in E^3. This Section is interested in representing the first order kinematics (i.e., the velocity) of the moving body.

4.2.1 Time derivative of pose—Derivative of homogeneous transform

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. (4.2):

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

The time derivative of this coordinate transformation is straightforward to calculate, given the time derivative of the rotation matrix, Eq. (3.20):

$$ \begin{aligned} {}_a\dot{\boldsymbol{p}} &= {}^b_a\dot{\boldsymbol{R}}\ {}_b\boldsymbol{p} + {}_a\dot{\boldsymbol{p}}^{a,b}, && (4.7)\\ &= {}^b_a\dot{\boldsymbol{R}}\ \left({}^b_a\boldsymbol{R}^T({}_a\boldsymbol{p} - {}_a\boldsymbol{p}^{a,b})\right) + {}_a\dot{\boldsymbol{p}}^{a,b}, \\ &= [{}_a\boldsymbol{\omega}]\ {}_a\boldsymbol{p} + {}_a\dot{\boldsymbol{p}}^{a,b} - [{}_a\boldsymbol{\omega}]\ {}_a\boldsymbol{p}^{a,b}. && (4.8) \end{aligned} $$

$\boldsymbol{\omega}$ 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. (3.19). Hence:

$$ \begin{pmatrix} {}_a\dot{\boldsymbol{p}} \\ 0 \end{pmatrix} = {}^b_a\dot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1} \begin{pmatrix} {}_a\boldsymbol{p} \\ 1 \end{pmatrix}, \quad \text{with} \quad {}^b_a\dot{\boldsymbol{T}} = \begin{pmatrix} {}^b_a\dot{\boldsymbol{R}} & {}_a\dot{\boldsymbol{a}}^{a,b} \\ \boldsymbol{0}_{1\times 3} & 0 \end{pmatrix}. \tag{4.9} $$

(Compare to Eq. (3.17).) In Eq. (4.9), the operator $\dot{\boldsymbol{T}}\ \boldsymbol{T}^{-1}$ works linearly on the coordinates of the point fixed to the moving body. Equation (4.8) can also be written as

$$ {}_a\dot{\boldsymbol{p}} = [{}_a\boldsymbol{\omega}]\ {}_a\boldsymbol{p} + {}_a\boldsymbol{v}_0, \tag{4.10} $$

with ${}_a\boldsymbol{v}_0 = {}_a\dot{\boldsymbol{p}}^{a,b} - [{}_a\boldsymbol{\omega}]\ {}_a\boldsymbol{p}^{a,b}$ the velocity of the point on the moving body that *instantaneously coincides* with the *origin* of the reference frame $\{a\}$ (Fig. 4.2). This yields a relationship that is similar to the corresponding relationship Eq. (3.20) for rotations:

$$ {}^b_a\dot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1} = \begin{pmatrix} [{}_a\boldsymbol{\omega}] & {}_a\boldsymbol{v}_0 \\ \boldsymbol{0}_{1\times 3} & 0 \end{pmatrix}. \tag{4.11} $$

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. (4.10) doesn't have a solution $\boldsymbol{p}$ for $\dot{\boldsymbol{p}} = 0$, since the coefficient matrix of $\boldsymbol{p}$ is not of full rank.

4.2.2 Time derivative of pose—(Velocity) twist

$\dot{\boldsymbol{T}}\,\boldsymbol{T}^{-1}$ in Eq. (4.11) is a 4×4 matrix, and hence it is a *non-minimal* representation of a given instantaneous velocity of a body that is located at a given pose at a given moment in time. The same complete motion information can be represented in a *minimal* representation, i.e., *two* three-dimensional coordinate vectors, $\boldsymbol{\omega}$ and $\boldsymbol{v}_0$, often assembled together in a six-dimensional "twist" coordinate vector $\mathbf{t}$:

$$ \mathbf{t} = \begin{pmatrix} \boldsymbol{\omega} \\ \boldsymbol{v}_0 \end{pmatrix}. \tag{4.12} $$

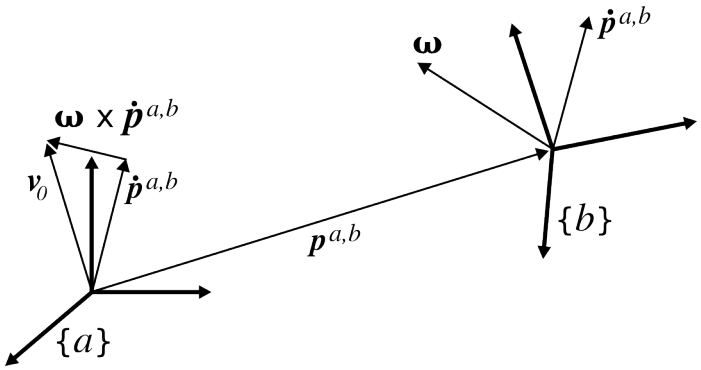

Figure 4.2: $\boldsymbol{v}_0$ is the velocity of the point of the moving rigid body that instantaneously coincides with the origin of the world frame $\{a\}$. It is the sum of (i) the translational velocity of the origin of the moving frame $\{b\}$, and (ii) the translational velocity generated by the angular velocity $\boldsymbol{\omega}$ working on the moving body at a distance $\boldsymbol{p}^{a,b}$ from the origin.

In this representation, adding rigid body velocities corresponds to adding twist vectors.

The order of the angular and linear velocity three-vectors in Eq. (4.12) is a completely arbitrary choice of this text. Some other publications use the alternative ordering choice:

$$\mathbf{t} = \begin{pmatrix} \boldsymbol{v}_0 \\ \boldsymbol{\omega} \end{pmatrix}. \tag{4.13}$$

A *second* six-dimensional coordinate vector alternative (with *different* physical interpretation!) for representing rigid body velocity is extracted from $\dot{\boldsymbol{T}}$ in Eq. (4.9):

$$\mathbf{t} = \begin{pmatrix} \boldsymbol{\omega} \\ \dot{\boldsymbol{p}}^{a,b} \end{pmatrix}. \tag{4.14}$$

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

> **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. (4.14). The linear velocity three-vector in a *screw twist* represents the velocity of the point that instantaneously coincides with the origin of $\{a\}$, Eq. (4.12).
>
> 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, because most engineers find them more intuitive (e.g., in the context of rigid body dynamics, Sect. 4.12.6). But "screw twists" are more informative than "pose twists":

> **Screw twist vs. Pose twist (cont'd)**: from *only* the six numbers in the screw twist, one can deduce *both* the position and orientation of the instantaneous screw axis; this is not possible with 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 *instantaneous screw axis* (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 three-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.

4.2.3 Transformation of twist coordinates

This Section extends Sect. 2.7 with the transformations of the coordinate representations of twists under a change of reference frame. Besides their obvious use in computer implementations, an important secondary motivation to study these transformations between different coordinate representations is the principle of *invariance*: theoretical and/or practical results derived in the framework of one particular coordinate representation should describe the same physical objects when transformed into another coordinate representation. This statement might seem trivial, but nevertheless many publications in the robotics literature violate this principle, [29, 71, 146].

Screw twists transform just like normal screws, Sect. 2.7.3, hence their name. This transformation, with the screw transformation matrix of Eq. (2.27), is repeated here for convenience:

$${}_f\mathbf{t} = {}^i_f\boldsymbol{S}\ {}_i\mathbf{t} = \begin{pmatrix} {}^i_f\boldsymbol{R} & \boldsymbol{0}_{3\times3} \\ \left[{}_f\boldsymbol{p}^{f,i}\right] {}^i_f\boldsymbol{R} & {}^i_f\boldsymbol{R} \end{pmatrix}\ {}_i\mathbf{t}. \tag{4.15}$$

Under a change of world reference frame, the three-vectors of a *pose* twist (Sect. 4.2.2) 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* ${}^i_f\boldsymbol{P}$:

$${}_f\mathbf{t} = {}^i_f\boldsymbol{P}\ {}_i\mathbf{t} = \begin{pmatrix} {}^i_f\boldsymbol{R} & \boldsymbol{0}_3 \\ \boldsymbol{0}_3 & {}^i_f\boldsymbol{R} \end{pmatrix}\ {}_i\mathbf{t}. \tag{4.16}$$

Just like ${}^i_f\boldsymbol{S}$, also ${}^i_f\boldsymbol{P}$ has always a unit determinant.

Inverse of pose twist transformation The inverse of ${}^i_f\boldsymbol{P}$ is trivial:

$$\left({}^i_f\boldsymbol{P}\right)^{-1} = \begin{pmatrix} {}^f_i\boldsymbol{R} & \boldsymbol{0}_3 \\ \boldsymbol{0}_3 & {}^f_i\boldsymbol{R} \end{pmatrix}. \tag{4.17}$$

Change of reference point on the moving body When the reference frame on the moving body changes, Fig. 2.6, 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:

$$\mathbf{t}^f = {}^i_f\boldsymbol{M}\ \mathbf{t}^i = \begin{pmatrix} \boldsymbol{I}_3 & \boldsymbol{0}_3 \\ \left[\boldsymbol{p}^{f,i}\right] & \boldsymbol{I}_3 \end{pmatrix}\ \mathbf{t}^i. \tag{4.18}$$

The trailing superscript indicates the reference point for the pose twist. Equation (4.18) is valid with respect to any world reference frame in which the coordinates of the twists and vectors are expressed.

4.3 Forces on a rigid body—Wrench

A *point mass* can be subjected to linear forces only (represented by three-dimensional vectors $\boldsymbol{f}$), while a *rigid body* can be subjected to both forces $\boldsymbol{f}$ and moments (or *"torque"*) $\boldsymbol{m}$. Such a combination $\boldsymbol{F} = (\boldsymbol{f}, \boldsymbol{m})$ of force and moment is called a *generalized force*, or sometimes also a *wrench*. This text often uses the same notation for the physical wrench and for its coordinate representation: $\boldsymbol{F} = (\boldsymbol{f}^T, \boldsymbol{m}^T)^T$.

Poinsot's Theorem (Sect. 4.8) says that every set of wrenches is equivalent to one single force and one single moment applied at a given *(force) reference point* on the rigid body. Wrenches even have the mathematical properties of *screws*, Sect. 2.6: one can find a representation where the physical force and moment vectors are lying on the same line.

Hence, wrenches can be represented by six-dimensional coordinate (column) vectors that are very similar to those of velocity twists. As in the case of twists, there are a number of arbitrary choices involved in the selection of wrench *coordinates*: the *reference point* at which the moment component of the wrench is taken; the physical units with which forces (Newton, pounds, ...) and moments (Newton-meter, Newton-millimeter, ...) are represented; the order of the force and moment components in the wrench coordinate vector; the frame in which the coordinates are expressed.

If a force $\boldsymbol{f}_1$ and moment $\boldsymbol{m}_1$ are known at a reference point "1," it is easy to find a physically equivalent set $(\boldsymbol{f}_2, \boldsymbol{m}_2)$ at another reference point "2":

$$\boldsymbol{f}_2 = \boldsymbol{f}_1, \qquad \boldsymbol{m}_2 = \boldsymbol{m}_1 + \boldsymbol{r}^{2,1} \times \boldsymbol{f}_1, \tag{4.19}$$

where $\boldsymbol{r}^{2,1}$ is the vector from "2" to "1." The force component is not influenced by the choice of reference point, but the moment component is. Equation (4.19) relates *physical* three-dimensional vectors, independent of the reference frame used to express the coordinates of the vectors. The relationship between the *coordinates* of the force/moment pairs at both reference points, expressed with respect to frames with origins at the reference points, is given by the *screw transformation matrix* that we know already from the coordinate transformation of twists, Eq. (4.15):

$$\begin{pmatrix} \boldsymbol{f}_f \\ \boldsymbol{m}_f \end{pmatrix} = {}_f\boldsymbol{F} = {}^i_f\boldsymbol{S}\ {}_i\boldsymbol{F} = \begin{pmatrix} {}^i_f\boldsymbol{R} & \boldsymbol{0}_{3\times 3} \\ \left[{}_f\boldsymbol{p}^{f,i}\right] {}^i_f\boldsymbol{R} & {}^i_f\boldsymbol{R} \end{pmatrix} \begin{pmatrix} \boldsymbol{f}_i \\ \boldsymbol{m}_i \end{pmatrix} \tag{4.20}$$

Recall that the inverse of the transformation is very cheap to calculate, Eq. (2.29). The *order* of the force and moment three-dimensional vectors $\boldsymbol{f}$ and $\boldsymbol{m}$ in Eq. (4.19) is *arbitrary*. Making the alternative choice, $\widetilde{\boldsymbol{F}} = (\boldsymbol{m}, \boldsymbol{f})$ implies that the coordinate transformation formulae have to be changed accordingly:

$${}_f\widetilde{\boldsymbol{F}} = \begin{pmatrix} \boldsymbol{m}_f \\ \boldsymbol{f}_f \end{pmatrix} = \boldsymbol{\Delta} \begin{pmatrix} \boldsymbol{f}_f \\ \boldsymbol{m}_f \end{pmatrix} = \boldsymbol{\Delta}\ {}^i_f\boldsymbol{S}\ {}_i\boldsymbol{F} = \boldsymbol{\Delta}\ {}^i_f\boldsymbol{S}\ \boldsymbol{\Delta}\ {}_i\widetilde{\boldsymbol{F}} = {}^i_f\boldsymbol{S}^S\ {}_i\widetilde{\boldsymbol{F}}, \tag{4.21}$$

with $\boldsymbol{\Delta}$ as in Eq. (2.19), and $\boldsymbol{S}^S$ the spatial transpose of $\boldsymbol{S}$, Eq. (2.31).

4.4 (Non-)invariants of rigid body motion and force

The previous Sections have introduced various *coordinate representations*, but one should not lose sight of the fact that they all describe the same *physical* motion or force. Hence, they all have the same so-called *(geometrical) invariants*. Physically, the finite or instantaneous motion of a rigid body is represented by: (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).

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 moment felt in a point on the screw axis, and (4) the pitch which is the ratio of the moment and force vectors.

Different authors use a different, arbitrary *order* of the three-vectors in the coordinate representations of twists (*screw* twists, *pose* twists, and *body-fixed* twists) and wrenches. The arbitrarily chosen order is, of course, not a *physical property* (or invariant) of the rigid body twists. This text uses the conventions introduced in Sects 4.2–4.3, because, with this choice, (screw) twists and (screw) wrenches transform in exactly the same way, while the alternative representation requires different transformation matrices for twists and wrenches. This could,

however, also have an advantage: twists and wrenches *are* two different things, and using different transformation equations for both emphasizes this difference.

4.5 Exponential and logarithm

Section 3.10 introduced the concept of the *exponentiation* for orientations. The same mapping exists from a twist (i.e., a rigid body velocity) onto a finite displacement (i.e., a pose): $\exp : \mathbf{t} \mapsto \boldsymbol{T}$. Equation (3.21) 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. (4.11), as the time derivative of rotation matrices, Eq. (3.20), a similar exponentiation formula works for twists and displacements too: the matrix exponential of the matrix corresponding to a *screw* twist $\mathbf{t} = ({}_a\boldsymbol{\omega}^T \; {}_a\boldsymbol{v}_0^T)^T = (\omega_x \, \omega_y \, \omega_z \, v_x \, v_y \, v_z)^T$ is the pose $\boldsymbol{T}$:

$$\boldsymbol{T} = \exp \begin{pmatrix} [{}_a\boldsymbol{\omega}] & {}_a\boldsymbol{v}_0 \\ \mathbf{0}_{1\times 3} & 0 \end{pmatrix}. \tag{4.22}$$

This works for *screw* twists only, not for pose twists.

The *logarithm of a finite displacement* is also a well-defined operation, that can be given a *physical* interpretation, [173, 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: the motion starts at the origin of the reference frame with respect to which the finite displacement is taken, and applies the same twist during one unit of time, to end up at the given finite displacement. When using a homogeneous transformation matrix for the finite displacement, the logarithm of this matrix gives the screw twist in the form of the argument of the exponential function in Eq. (4.22).

4.5.1 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*, [107, 173, 229], 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. (This is exactly the interpretation of the previous Section.)

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

$$\begin{aligned} \boldsymbol{T} = &\exp\left((\omega'_x \; 0 \; 0 \; 0 \; 0 \; 0)^T\right) \\ &\exp\left((0 \; \omega'_y \; 0 \; 0 \; 0 \; 0)^T\right) \\ &\exp\left((0 \; 0 \; \omega'_z \; 0 \; 0 \; 0)^T\right) \\ &\exp\left((0 \; 0 \; 0 \; v'_x \; 0 \; 0)^T\right) \\ &\exp\left((0 \; 0 \; 0 \; 0 \; v'_y \; 0)^T\right) \\ &\exp\left((0 \; 0 \; 0 \; 0 \; 0 \; v'_z)^T\right). \end{aligned} \tag{4.23}$$

This is an example of the composition of transformation matrices, Eq. (4.4), 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):

$$\boldsymbol{T} = \boldsymbol{R}(X, \omega'_x)\boldsymbol{R}(Y, \omega'_y)\boldsymbol{R}(Z, \omega'_z)\boldsymbol{Tr}(X, v'_x)\boldsymbol{Tr}(Y, v'_y)\boldsymbol{Tr}(Z, v'_z), \tag{4.24}$$

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.)

4.6 Reciprocity of twists and wrenches

The reciprocal product of screws (Sect. 2.6.2) directly applies twists and wrenches too. A twist represents a motion of a rigid body; a wrench represents a force on a rigid body. The *work* that a moving rigid body generates when it moves under a given force is simply the "product" of the motion with the force (remember the *force times displacement* in the case of a particle moving under a linear force). For twists that represent *instantaneous velocities*, "work" is replaced by its instantaneous equivalent "power." This power relationship between a twist $\mathbf{t}$ a wrench $\mathbf{F}$ is described by the *spatial scalar product* or *reciprocal product*:

$$P(\mathbf{t}, \mathbf{F}) = \boldsymbol{\omega} \cdot \boldsymbol{m} + \boldsymbol{v} \cdot \boldsymbol{f}, \tag{4.25}$$

with $\boldsymbol{\omega}$ the angular velocity three-vector, $\boldsymbol{v}$ the linear velocity three-vector, $\boldsymbol{f}$ the linear force three-vector, and $\boldsymbol{m}$ the moment of force three-vector. It is clear that the P is *bilinear*:

$$P(k^1\mathbf{t}^1 + k^2\mathbf{t}^2, \mathbf{F}) = k^1 P(\mathbf{t}^1, \mathbf{F}) + k^2 P(\mathbf{t}^2, \mathbf{F}), \tag{4.26}$$

$$\text{and} \quad P(\mathbf{t}, k^1\mathbf{F}^1 + k^2\mathbf{F}^2) = k^1 P(\mathbf{t}, \mathbf{F}^1) + k^2 P(\mathbf{t}, \mathbf{F}^2). \tag{4.27}$$

In robotics, one often needs to calculate the reciprocal set of a set of screws. For example, a five degrees-of-freedom robot (i.e., a robot that can generate velocities for its end-effector in five independent directions) 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, without requiring any *power* from the motors. Another example are the constraint forces in ideal (frictionless, infinitely stiff) *motion constraints* on a rigid body: all constraint forces that can be generated in such a constraint generate *no power* against all velocities that the body can have while satisfying the constraints. In fact, both examples above illustrate the same physical property, since the joints in the robot example are assumed to be ideal motion constraints on the robot's end-effector body.

Reciprocity of twists and wrenches *is not* the same thing as orthogonality, since twists and wrenches live in different physical space, while orthogonality is only defined between members of the same space. Some literature chooses the coordinate representations of twists and wrenches in such a way as to make reciprocity and orthogonality *look* the same:

$$P(\mathbf{t}, \mathbf{F}) = \boldsymbol{\omega} \cdot \boldsymbol{m} + \boldsymbol{v} \cdot \boldsymbol{f} = \mathbf{t}^T \mathbf{F} = 0. \tag{4.28}$$

4.7 Twists and wrenches as elements of a vector space

The reciprocal product between twists and wrenches, Eq. (4.25), is the last of the three important parallels between the algebraic properties of the Euclidean three-vector space and the screw six-vector space:

1. Screws or twists are composed by simple *addition*, just like Euclidean three-vectors.
2. The *reciprocal product* of twists is a generalization of the scalar product of three-vectors. It has the same algebraic properties, and can also be interpreted as a *projection*.
3. Similarly, the *motor product* of twists (Sect. 4.10.2) is a generalization of the cross product of three-vectors. It has the same algebraic properties, and can also be interpreted as a *motion operator*, i.e., it transforms two motions into a third one.

4.8 Duality of twists and wrenches

Twist and wrenches are both "built on top of" the same geometrical concept of the screw, Sect. 2.6. 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 *dual* result for the wrenches built with the same screws. Such dualities occur very often in the motion and force analysis of kinematic chains. The following paragraphs give one famous example, in the form of two important theorems from the 19th century. The first is from the French mathematician Michel Chasles (1793–1881), [41]:

> **Chasles' Theorem, 1830.** The most general finite displacement motion for a rigid body is a *screw motion*, [5, 15, 24, 84], i.e., there exists a line in space (called the "screw axis" (SA), Sect. 2.6, [15, 121, 220], or "twist axis") such that the body's motion is a rotation about the SA plus a translation along it.

The proof of this theorem is simple when using coordinates, and a constructive way of finding the screw axis is given in Sect. 2.6.1. Chasles himself formulated his principle in many different ways. The one that comes closest to the modern formulation is probably the following, [41, 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*. The notion of twist axis was probably already discovered many years before Chasles (the earliest reference seems to be the Italian Giulio Mozzi (1763), [40, 82, 170]) but he normally gets the credit.

A second similar theorem deals with the six-dimensional forces, or "*wrenches*", that can act on a rigid body. These forces also have a screw axis, as was first formulated by the French geometer Louis Poinsot (1777–1859) [206]:

> **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.

"An angular moment in a plane perpendicular to the force" is equivalent to a moment vector in the same direction as the force; hence, the screw axis also exists for force. The proof of Poinsot's Theorem runs along exactly the same lines as that of Chasles, since both twists and wrenches are instantiations of the same geometric concept of a screw.

4.9 Metric properties of twists and wrenches

This Section explains the differences between the six-dimensional space of twists and the three-dimensional Euclidean space E^3, with respect to the definition of norms or magnitudes. The former space has *no natural distance* function. This would mean that an obvious (i.e., physically meaningful) distance function exists, that applies to all problems in the space it is defined on. For example, the Euclidean metric is a natural choice for the space of *points*. For twists and wrenches, however, such a natural choice does not exist, [150, 173, 193]. (It *does* exist when one considers rotations only, Sect. 3.22.) The result of this absence of natural metric is that any "norm" that is being used must, in one way or another, make an *arbitrary choice* of "weighting factor" w between translation and orientation:

$$||\mathbf{t}||_W = \mathbf{t}^T \boldsymbol{W} \mathbf{t} = (\boldsymbol{\omega}^T \ \boldsymbol{v}^T) \boldsymbol{W} \begin{pmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{pmatrix}, \tag{4.29}$$

where $\boldsymbol{W}$ is the 6×6 matrix that is the mathematical representation of the weighting. All elements in this matrix do not have the same physical dimensions!

Probably the most important practical consequence of this fact is that one should be *very* careful when reading about, or developing yourself, "optimal" solutions to robotics problems: the optimum is always defined with respect to an (often implicitly!) chosen metric.

4.10 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.

4.10.1 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. (4.8):

$${}_a\dot{\boldsymbol{p}} = [{}_a\boldsymbol{\omega}] \ {}_a\boldsymbol{p} + {}_a\dot{\boldsymbol{p}}^{a,b} - [{}_a\boldsymbol{\omega}] \ {}_a\boldsymbol{p}^{a,b}.$$

Taking the time derivative of both sides and using Eq. (4.9) gives

$$\begin{aligned} {}_a\ddot{\boldsymbol{p}} &= [{}_a\dot{\boldsymbol{\omega}}]\ {}_a\boldsymbol{p} + [{}_a\boldsymbol{\omega}]\ {}_a\dot{\boldsymbol{p}} + {}_a\ddot{\boldsymbol{p}}^{a,b} - [{}_a\dot{\boldsymbol{\omega}}]\ {}_a\boldsymbol{p}^{a,b} - [{}_a\boldsymbol{\omega}]\ {}_a\dot{\boldsymbol{p}}^{a,b} \\ &= [{}_a\dot{\boldsymbol{\omega}}]\ {}_a\boldsymbol{p} + [{}_a\boldsymbol{\omega}]\ [{}_a\boldsymbol{\omega}]\ {}_a\boldsymbol{p} + {}_a\ddot{\boldsymbol{p}}^{a,b} - [{}_a\dot{\boldsymbol{\omega}}]\ {}_a\boldsymbol{p}^{a,b} - [{}_a\boldsymbol{\omega}]\ [{}_a\boldsymbol{\omega}]\ {}_a\boldsymbol{p}^{a,b} \\ &= ([{}_a\dot{\boldsymbol{\omega}}] + [{}_a\boldsymbol{\omega}]\ [{}_a\boldsymbol{\omega}])\ {}_a\boldsymbol{p} + {}_a\boldsymbol{a}_0, \end{aligned} \tag{4.30}$$

with ${}_a\boldsymbol{a}_0 = \dot{\boldsymbol{v}}_0 + [{}_a\boldsymbol{\omega}]\ {}_a\boldsymbol{v}_0$, and ${}_a\boldsymbol{v}_0$ as in Eq. (4.10). 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*, [6, 84]. In general, the determinant of the coefficient matrix of $\boldsymbol{p}$ in Eq. (4.30) does *not* vanish, such that a point with instantaneous vanishing acceleration exists: solve for ${}_a\boldsymbol{p}$ from Eq. (4.30), with left-hand side equal to zero. This point is often called the *acceleration centre*, or *acceleration pole*, [21, 22, 110, 237, 239]. The matrix form of Eq. (4.30) is straightforwardly found from Eq. (4.9), [252]:

$$\begin{aligned} \begin{pmatrix} {}_a\ddot{\boldsymbol{p}} \\ 0 \end{pmatrix} &= \frac{d}{dt}\left({}^b_a\dot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1}\right)\begin{pmatrix} {}_a\boldsymbol{p} \\ 1 \end{pmatrix} + {}^b_a\dot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1}\begin{pmatrix} {}_a\dot{\boldsymbol{p}} \\ 0 \end{pmatrix} \\ &= \left({}^b_a\ddot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1} + {}^b_a\dot{\boldsymbol{T}}\ \frac{d}{dt}({}^b_a\boldsymbol{T}^{-1}) + {}^b_a\dot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1}\ {}^b_a\dot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1}\right)\begin{pmatrix} {}_a\boldsymbol{p} \\ 1 \end{pmatrix} \\ &= \left({}^b_a\ddot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1} - {}^b_a\dot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1}{}^b_a\dot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1} + {}^b_a\dot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1}{}^b_a\dot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1}\right)\begin{pmatrix} {}_a\boldsymbol{p} \\ 1 \end{pmatrix}. \end{aligned} \tag{4.31}$$

or

$$\begin{pmatrix} {}_a\ddot{\boldsymbol{p}} \\ 0 \end{pmatrix} = {}^b_a\ddot{\boldsymbol{T}}\ {}^b_a\boldsymbol{T}^{-1}\begin{pmatrix} {}_a\boldsymbol{p} \\ 1 \end{pmatrix}, \quad \text{with} \quad {}^b_a\ddot{\boldsymbol{T}} = \begin{pmatrix} ([\dot{\boldsymbol{\omega}}] + [\boldsymbol{\omega}][\boldsymbol{\omega}])\,{}^b_a\boldsymbol{R} & {}_a\ddot{\boldsymbol{p}}^{a,b} - [\boldsymbol{\omega}]_a\dot{\boldsymbol{p}}^{a,b} - [\dot{\boldsymbol{\omega}}]_a\dot{\boldsymbol{p}}^{a,b} \\ \boldsymbol{0}_{1\times 3} & 0 \end{pmatrix}. \tag{4.32}$$

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. 4.2.2), the information contained in $\ddot{\boldsymbol{T}}\boldsymbol{T}^{-1}$ *cannot* be reduced to two three-vectors, Sect. 4.2.1: it contains linear and angular velocity three-vectors, as well as their time derivatives, so *four independent three-vectors* in total.

Nevertheless, two *six-vector* "acceleration" descriptions are in common use, especially in fluid mechanics, [247, p. 227] [158, p. 433]. In fluid mechanics, this is a valid approach since one considers *moving point masses* and not *moving rigid bodies*; applying these "accelerations" to moving bodies, however, inevitably gives rise to confusion and incorrect interpretations sooner or later! Both six-vector acceleration descriptions differ in the choice of the particular reference point whose acceleration they represent:

1. *Screw acceleration* twist (*Eulerian* or ordinary time derivative; *acceleration motor*, [25, p. 127]; *spatial acceleration*, [76]). This is the three-vector ${}_a\boldsymbol{v}_0$ in Eq. (4.30). It represents the change in the velocity three-vector of the vector field at a point that coincides with the reference point which is fixed in space. That means that the acceleration is the difference between the velocities of *two different* particles, at the *same place* in space but at *different instants in time*.

2. *Pose acceleration* twist (*Lagrangian* or *material* derivative): one follows the change in the velocity of one single particle that moves with the rigid body. The acceleration is the difference between the velocities of the *same particle* at *two different instances in time*, and hence also at two *different places* in space.

Note that for velocity twists, the Eulerian time derivative equals the Lagrangian time derivative. Important in this Section is that rigid body acceleration is not a screw vector. The acceleration of a moving rigid body *cannot* be represented in one single six-vector. This does not contradict the fact that the *screw acceleration* twist *is* a screw: the acceleration six-vectors of the previous paragraphs just don't represent *all of* the body's acceleration.

There is one special case in which the time derivative of the velocity screw *does* give a correct acceleration: if the body is momentarily in rest, both $\boldsymbol{v}_0$ and $\boldsymbol{\omega}$ are zero, and $\dot{\mathbf{t}} = (\dot{\boldsymbol{v}}_0^T\ \dot{\boldsymbol{\omega}}^T)^T$ contains all acceleration information.

4.10.2 Motor product—Derivative of screw along a 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 $\boldsymbol{S}_\Delta(\mathbf{t}_\Delta = \Delta t\,\mathbf{t}^1)$, Sect. 2.33. Hence, the time derivative of $\mathbf{t}^2$ is found as

$$\begin{aligned}\frac{d\mathbf{t}^2}{dt} &= \lim_{\Delta t\to 0}\frac{\boldsymbol{S}_\Delta(\Delta t\,\mathbf{t}^1)\mathbf{t}^2 - \mathbf{t}^2}{\Delta t} \\ &= \begin{pmatrix}[\boldsymbol{\omega}^1] & \boldsymbol{0}\\ [\boldsymbol{v}^1] & [\boldsymbol{\omega}^1]\end{pmatrix}\mathbf{t}^2 \qquad (4.33)\\ &= \mathbf{t}^1 \times \mathbf{t}^2. \qquad (4.34)\end{aligned}$$

This last relationship is *motor product* (Sect. 2.6.3) or *Lie bracket* of both screws. It is tempting to consider Eq. (4.34) as the definition of the acceleration of a body moving with a twist $\mathbf{t}$. However, this is so only in a couple of special cases; the following Section explains why.

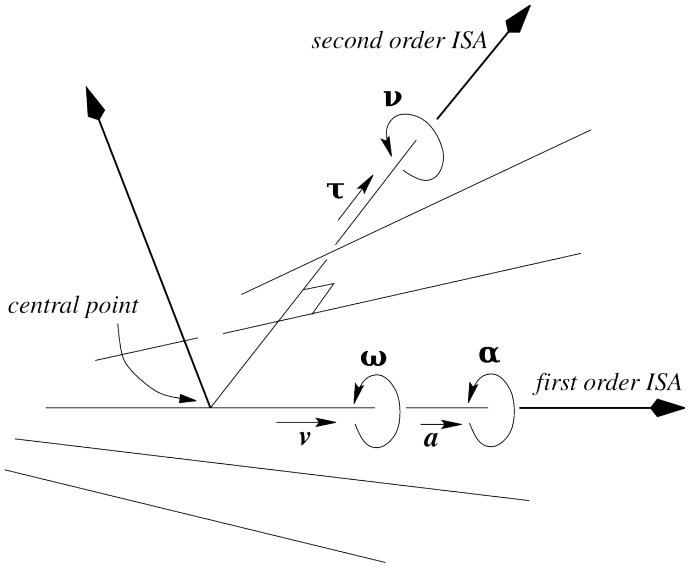

Figure 4.3: First order screw axes (ISA) for a general rigid body motion at different instants in time. The common normal between two subsequent ISAs is the second order screw axis.

4.10.3 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, [22, 239, 238]. These two six-vectors represent two *instantaneous screw axes*, Sect. 4.1.8: (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. 4.3). 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, [24, p. 158], [121, 214, 237], [252, 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 twelve 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 six 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 two extra components for these two vectors, since one knows that they lie on the ISA.)

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 two 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 two more parameters, representing their magnitudes; their direction is already determined by the second order ISA.)

4.11 Point mass dynamics

Before tackling the dynamics of a complete rigid body (in Sect. 4.12), this Section first describes the dynamics of one single, ideal point mass. All concepts introduced here will generalize smoothly to rigid bodies, and to kinematic chains (in Chap. 6). In his *Principia Mathematica* (first published in 1687, [186]), Sir Isaac Newton (1642–1727) stated the three physical principles behind the dynamics theory required in this text:

1. *Law of inertia.* A point mass, on which no forces act, continues in a rectilinear motion with constant velocity. This velocity is zero for a point mass in rest.

2. *Equation of motion.* The relation between a point mass' acceleration $\ddot{\boldsymbol{r}}$ and the force $\boldsymbol{f}$ exerted on it, is *linear*. The proportionality factor is the *mass* m:

$$\boldsymbol{f} = m\ddot{\boldsymbol{r}}. \tag{4.35}$$

3. *Action and reaction.* The forces that two bodies exert on each other are equal and opposite.

Newton discussed only *point masses*, such that his laws apply only directly to the *translational motion* of the *centre of mass* of a rigid body. Hence, the motion parameters of the rigid body are restricted to E^3: its acceleration three-vector $\ddot{\boldsymbol{r}}$, and the linear force three-vector $\boldsymbol{f}$ exerted at the centre of mass.

Equation (4.35) is Newton's second law in its most widely used form: it assumes that the mass m does not change over time. However, this law has an alternative formulation that is more general, in that it allows the inertial properties (i.e., mass) to change too:

$$\boldsymbol{f} = \frac{d(m\dot{\boldsymbol{r}})}{dt} = \frac{d\boldsymbol{p}}{dt}. \tag{4.36}$$

The three-vector $\boldsymbol{p}$ is the *linear momentum* of the moving body. (For compatibility reasons with most dynamics literture, this Chapter uses a little different notation than the other Chapters: the symbol $\boldsymbol{p}$ denotes a momentum, and $\boldsymbol{r}$ denotes a position.) Unlike the velocity three-vector $\dot{\boldsymbol{r}}$, the linear momentum three-vector $\boldsymbol{p}$ is a *line* vector, just like the force three-vector to which it is linked by Eq. (4.36). From this relationship it is clear that the *structural* properties of momentum are the same as those of force. The law of *conservation of linear momentum* is a direct consequence of the momentum equation (4.36): if no external forces work on the mass, the linear momentum does not change over time.

The time invariance of the body's mass may seem obvious for a rigid body, but for a kinematic *chain* of rigid bodies it isn't: since the rigid bodies that make up the manipulator can move with respect to each other, the inertial properties of the total manipulator (such as its centre of mass) do change over time. So, instead of the acceleration equation Eq. (4.35), it will be easier to generalise the momentum equation Eq. (4.36) from the case of one single rigid body to the case of an articulated chain of rigid bodies.

4.11.1 Coriolis and centrifugal effects

Newton's equation of motion, Eq. (4.35), assumes that the three-vectors $\boldsymbol{f}$ and $\ddot{\boldsymbol{r}}$ are expressed with respect to an *inertial frame*. The formal definition of such an inertial frame seems to give rise to a cyclic reasoning: an inertial frame is a frame in which Newton's law is valid. The choice of an appropriate inertial frame is task-dependent: for most practical applications, a frame fixed to the earth suffices; celestial or global dynamics require frames attached to the solar system or even the "fixed stars." In robotics, the base frames of robots that are fixed to the earth are most often valid inertial frames, because the earth's dynamic effects (i.e., caused by the difference in distances to the earth's graviational centre) are orders of magnitude smaller than the dynamics of the robot and its task. But frames connected to the end-effector, or to one or more of the intermediate links of the robots, are typically not inertial frames, because they move with speeds and accelerations comparable to those of the robot's task. For many robots, however, it is exactly in these moving frames that one has to express the dynamics of the robot, since the motors generate the forces and torques in these frames. Hence, it is necessary to know the expression of Newton's law in a moving reference frame. This expression *formally* gets extra terms, in addition

to the "$\boldsymbol{f} = m\boldsymbol{a}$" component. However, these extras are not physical forces, but artefacts of the coordinate representations in moving frames. The reason they are not physical is that you can not use them to generate useful power to perform a task, for example.

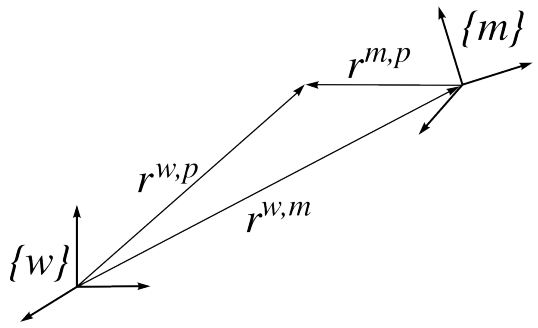

Figure 4.4: Position vector triangle for a point p moving with respect to a frame $\{m\}$ that itself moves with respect to an inertial world reference frame $\{w\}$.

So, imagine a point with mass m that moves with respect to a reference frame $\{m\}$, which itself moves with respect to an inertial world reference frame $\{w\}$. The question is now: *"How does an observer moving together with the frame $\{m\}$ see the relationship between the acceleration ${}_m\ddot{\boldsymbol{r}}$ of the moving point mass as he measures it in his frame $\{m\}$, and the force ${}^f\boldsymbol{m}$ exerted on the point mass?"* The answer to this question starts from the position vector triangle in Fig. 4.4. Its coordinate representation with respect to the world reference frame $\{w\}$ is:

$$ {}_w\boldsymbol{r}^{w,p} = {}_w\boldsymbol{r}^{w,m} + {}_w\boldsymbol{r}^{m,p} = {}_w\boldsymbol{r}^{w,m} + {}^m_w\boldsymbol{R}\ {}_m\boldsymbol{r}^{m,p}. \tag{4.37} $$

The index "p" refers to the position of the point mass. The time derivative of this identity is calculated as in Sect. 4.2.1, but now with time-varying coordinates of the point with respect to the moving frame. A simple extension of Eq. (4.8), [84, p. 176], gives

$$ {}_w\dot{\boldsymbol{r}}^{w,p} = {}_w\dot{\boldsymbol{r}}^{w,m} + {}_w\dot{\boldsymbol{r}}^{m,p} + [{}_w\boldsymbol{\omega}]\ {}_w\boldsymbol{r}^{m,p}. \tag{4.38} $$

A second time derivation, and the application of Eq. (3.20), yield the acceleration of the point mass with respect to $\{w\}$:

$$ {}_w\ddot{\boldsymbol{r}}^{w,p} = {}_w\ddot{\boldsymbol{r}}^{w,m} + {}_w\ddot{\boldsymbol{r}}^{m,p} + ([{}_w\dot{\boldsymbol{\omega}}] + [{}_w\boldsymbol{\omega}]\,[{}_w\boldsymbol{\omega}])\ {}_w\boldsymbol{r}^{m,p} + 2[{}_w\boldsymbol{\omega}]\ {}_w\dot{\boldsymbol{r}}^{m,p}. \tag{4.39} $$

Hence, as seen from the inertial frame, the point mass accelerates due to several causes:

1. If the point mass is fixed to the moving frame $\{m\}$, this frame's linear acceleration ${}_w\ddot{\boldsymbol{r}}^{w,m}$ makes the point mass accelerate with it.

2. The point mass can accelerate as seen from $\{m\}$. This gives rise to a change ${}_w\ddot{\boldsymbol{r}}^{m,p}$ in both *direction* and *magnitude* of the velocity ${}_w\dot{\boldsymbol{r}}^{m,p}$ of the point mass.

3. If $\{m\}$ undergoes an angular acceleration $\dot{\boldsymbol{\omega}}$, the point mass accelerates too, hence the term $[{}_w\dot{\boldsymbol{\omega}}]_w\boldsymbol{r}^{m,p}$.

4. The angular velocity of $\{m\}$ induces an acceleration of the point mass, even if (i) $\{m\}$ has no inertial acceleration itself (i.e., ${}_w\ddot{\boldsymbol{r}}^{w,m} = \dot{\boldsymbol{\omega}} = 0$) or (ii) the point mass does not move with respect to $\{m\}$ (i.e., ${}_m\dot{\boldsymbol{r}}^{m,p} = {}_m\ddot{\boldsymbol{r}}^{m,p} = 0$). The term $[\boldsymbol{\omega}]\,[\boldsymbol{\omega}]\boldsymbol{r}^{m,p}$ is a three-vector acceleration, that points from the point mass *towards* the rotation axis. Hence, it is called the *centripetal acceleration* of the point mass. "Centripetal" literally means "proceeding or acting in a direction towards the centre."

5. If the point mass moves with respect to $\{m\}$, *and* $\{m\}$ rotates with respect to $\{w\}$, the point mass accelerates with the so-called *Coriolis acceleration* $2[\boldsymbol{\omega}]\,\dot{\boldsymbol{r}}^{m,p}$, [59]. The French civil engineer Gaspard G. de Coriolis (1792–1843) was the first to use this reasoning to explain why reference frames fixed to the rotating earth are not perfect inertial frames.

In the inertial world frame $\{w\}$, Newton's law connects the acceleration ${}_w\ddot{\boldsymbol{r}}$ of the point mass m to the force ${}_w\boldsymbol{f}$ working on it: ${}_w\boldsymbol{f} = m\ {}_w\ddot{\boldsymbol{r}}$. The observer, however, sees the point mass move with a velocity ${}_m\dot{\boldsymbol{r}}^{m,p}$ and with an

acceleration ${}_m\ddot{\boldsymbol{r}}^{m,p}$; while he sees the force ${}_m\boldsymbol{f} = {}^m_w\boldsymbol{R}\ {}_w\boldsymbol{f}$ exerted on the point. Hence, according to this moving observer, the relationship between both is

$$\boldsymbol{f} - m\,[\boldsymbol{\omega}]\,[\boldsymbol{\omega}]\,\boldsymbol{r} - 2m\,[\boldsymbol{\omega}]\,\dot{\boldsymbol{r}} = m\ddot{\boldsymbol{r}}. \tag{4.40}$$

(For simplicity, we omitted the subscript "m" and all other superscripts, and assumed that $\{m\}$ does not accelerate with respect to $\{w\}$.) This relationship means that the observer sees the same formal linear relation between force and acceleration ($\boldsymbol{f} = m\ddot{\boldsymbol{r}}$), but "disturbed" by two "forces": the *centrifugal force* $\boldsymbol{f}^{cf} = -m\,[\boldsymbol{\omega}]\,[\boldsymbol{\omega}]\,\boldsymbol{r}$, and the *Coriolis force* $\boldsymbol{f}^{co} = -2m\,[\boldsymbol{\omega}]\,\dot{\boldsymbol{r}}$. The centrifugal force $\boldsymbol{f}^{cf}$, as defined here, points *away* from the rotation axis, hence the adjective "centrifugal," which literally means "proceeding or acting in a direction away from the centre." If we define the disturbance as "$m\,[\boldsymbol{\omega}]\,[\boldsymbol{\omega}]\,\boldsymbol{r}$, then it is called the *centripetal* force, since it is directed *towards* the rotation axis. The centrifugal force $\boldsymbol{f}^{cf}$ vanishes if the point mass coincides with the origin of $\{w\}$, i.e., $\boldsymbol{r}^{m,p} = 0$. The Coriolis force $\boldsymbol{f}^{co}$ vanishes if the point mass doesn't move with respect to the moving frame $\{m\}$, i.e., ${}_m\dot{\boldsymbol{r}} = 0$. Both forces vanish if $\{m\}$ does not rotate with respect to $\{w\}$, i.e., $\boldsymbol{\omega} = 0$ and hence $\{m\}$ is also an inertial frame. It is important to understand that the centrifugal and Coriolis forces are *real* to the extent that an observer moving with the point mass will actually be able to feel or *measure* these forces. On the other hand, the centrifugal and Coriolis forces have no *physical origin* except that they are only "caused" by the fact that Newton's equation of motion, Eq. (4.35), does not hold *formally* for a moving observer. And centrifugal and Coriolis forces cannot be used to generate useful energy.

In summary, the *Coriolis* force is proportional to (i) the velocity of the mass with respect to the moving reference frame, and (ii) the angular velocity of this moving reference frame. The *centrifugal* force is proportional to the *square* of the angular velocity of the moving reference frame. Both forces depend linearly on the moving mass.

4.11.2 Moment of force—Moment of momentum

The *moment of force* $\boldsymbol{m}$ (also called "moment" for short) of the three-vector $\boldsymbol{f}$ that represents the action of a force on a line, is classically defined as the vector product of (i) the lever arm $\boldsymbol{r}$ between the origin of the chosen reference frame and the line of action of the force, and (ii) the force $\boldsymbol{f}$:

$$\boldsymbol{m} = \boldsymbol{r} \times \boldsymbol{f}. \tag{4.41}$$

Similarly, one defines the *moment of momentum* $\boldsymbol{l}$ of a moving point mass (with position vector $\boldsymbol{r}$, velocity $\dot{\boldsymbol{r}}$, and hence linear momentum $\boldsymbol{p} = m\dot{\boldsymbol{r}}$) as

$$\boldsymbol{l} = \boldsymbol{r} \times \boldsymbol{p} = \boldsymbol{r} \times m\dot{\boldsymbol{r}}. \tag{4.42}$$

$\boldsymbol{p}$ is a *line* vector, but $\boldsymbol{l}$ is a *free* vector. The moment of momentum is often also called the *angular momentum* of the point mass. Newton's equation of motion $\boldsymbol{f} = d\boldsymbol{p}/dt$ extends straightforwardly to

$$\boldsymbol{m} = \boldsymbol{r} \times \boldsymbol{f} = \boldsymbol{r} \times \frac{d\boldsymbol{p}}{dt}. \tag{4.43}$$

In the identity

$$\frac{d}{dt}(\boldsymbol{r} \times \boldsymbol{p}) = \frac{d\boldsymbol{r}}{dt} \times \boldsymbol{p} + \boldsymbol{r} \times \frac{d\boldsymbol{p}}{dt}, \tag{4.44}$$

the first term on the right-hand side vanishes because the linear momentum $\boldsymbol{p}$ is parallel to the time derivative of the position vector $\boldsymbol{r}$: both are proportional to the velocity $\dot{\boldsymbol{r}}$ of the moving mass. This gives the more familiar form of the moment form of Newton's law of motion:

$$\boldsymbol{m} = \frac{d\boldsymbol{l}}{dt} = \frac{d}{dt}(\boldsymbol{r} \times \boldsymbol{p}). \tag{4.45}$$

This identity gives also rise to an obvious conservation property: if no external torques work on the moving mass, then its angular momentum does not change over time.

4.11.3 Physical units

From the definitions (4.35), (4.41) and (4.42), the physical units of force $\boldsymbol{f}$, moment of force $\boldsymbol{m}$, and angular momentum $\boldsymbol{l}$ are seen to be:

$$[\boldsymbol{f}] = \text{mass} \times \text{acceleration} = \frac{\text{kg m}}{\text{s}^2} = \text{Newton (N) in SI units}, \tag{4.46}$$

$$[\boldsymbol{m}] = \text{force} \times \text{distance} = \frac{\text{kg m}^2}{\text{s}^2} = \text{Newton-meter (Nm) in SI units,} \tag{4.47}$$

and

$$[\boldsymbol{l}] = \text{distance} \times \text{mass} \times \text{velocity} = \frac{\text{kg m}^2}{\text{s}} = \text{Newton-meter-second (Nms) in SI units.} \tag{4.48}$$

4.12 Rigid body dynamics

This Section extends the Newtonian dynamics for a *point mass* (Sect. 4.11) to the Newtonian dynamics for a single rigid body, by taking orientation into account. The Swiss mathematician Leonhard Euler (1707–1783) first did this around 1735, describing the dynamics of rigid bodies that have one point fixed, such as spinning tops or gyroscopes; later authors attached his name to that of Newton, now speaking of the *Newton-Euler equations* for rigid body dynamics. The obvious generalizations soon followed: the fully unconstrained rigid body; the reformulation as seen from a reference frame that is itself moving in an arbitrary way; the (partially) *constrained* rigid body (Sect. **??**). All these generalizations are relevant in robotics: the motors in the joints of the robot must move the mass of all more distal parts of the robot, while they are themselves moved by the more proximal joints, *and* all segments in the robot's kinematic chain are constrained, by other segments or by physical interactions with objects in the environment.

4.12.1 Decoupling of rigid body dynamics about centre of mass

A rigid body can be looked at as a continuous distribution of point masses that are rigidly connected through cohesion forces $\boldsymbol{f}_{ij}$ and $\boldsymbol{f}_{ji}$ acting between each pair of points i and j in the body. ($\boldsymbol{f}_{ij}$ is the force exerted by point i on point j.) These forces $\boldsymbol{f}_{ij}$ and $\boldsymbol{f}_{ji}$ are equal in magnitude and opposite in direction:

$$\boldsymbol{f}_{ij} = -\boldsymbol{f}_{ji}. \tag{4.49}$$

Newton's equations of motion, Eqs (4.35) and (4.36), are only valid for each of these point masses individually. In these dynamical equations, the rigid body constraint forces $\boldsymbol{f}_{ji}$ act as *external* forces on the point mass i. Hence, they are to be included as such in Newton's equation of motion, Eq. (4.35):

$$\boldsymbol{f}_i^{\text{ext}} + \sum_j \boldsymbol{f}_{ji} = \frac{d\boldsymbol{p}_i}{dt}. \tag{4.50}$$

This equation describes the motion (i.e., the change in the linear momentum) of the ith particle, under the influence of (i) the external force $\boldsymbol{f}_i^{\text{ext}}$ acting on the point, and (ii) the rigidity constraint forces $\boldsymbol{f}_{ji}$ exerted by all other points j in the body. An equation like (4.50) holds for all points in the body. Hence, the summation of these equations over all points gives

$$\sum_i \boldsymbol{f}_i^{\text{ext}} + \sum_{ij} \boldsymbol{f}_{ji} = \sum_i \frac{d\boldsymbol{p}_i}{dt}. \tag{4.51}$$

The first term in the left-hand side of this equation is the total external force $\boldsymbol{f}$ that acts on the rigid body. The second term vanishes, because of the "action equals reaction" principle of Eq. (4.49). The right-hand side can be rewritten as

$$\begin{aligned}
\sum_i \frac{d\boldsymbol{p}_i}{dt} &= \sum_i \frac{d^2(m_i \boldsymbol{r}_i)}{dt^2} \\
&= \frac{d^2}{dt^2} \sum_i m \frac{m_i}{m \boldsymbol{r}_i} \quad \text{with} \quad m = \sum_i m_i && (4.52) \\
&= \frac{d^2(m\boldsymbol{r}^c)}{dt} \quad \text{with} \quad \boldsymbol{r}^c = \frac{1}{m} \sum_i m_i \boldsymbol{r}_i && (4.53) \\
&= \frac{d\boldsymbol{p}^c}{dt}. && (4.54)
\end{aligned}$$

$\boldsymbol{r}^c$ is the position vector of the so-called *centre of mass* of the set of particles. m is the total mass of the body. $\boldsymbol{p}^c$ is the linear momentum of a virtual point mass, carrying all mass m of the total rigid body and moving together

with the centre of mass of the rigid body. If the rigid body has a continuous mass distribution, the summation is replaced by an integral, and the mass of point i is replaced by the product of a mass distribution $\rho(\boldsymbol{r})$ with an infinitesimal volume dV at the position $\boldsymbol{r}$:

$$\boldsymbol{r}^c = \frac{1}{m} \int_V \rho(\boldsymbol{r}) \boldsymbol{r} \, dV, \tag{4.55}$$

and

$$m = \int_V \rho(\boldsymbol{r}) \, dV. \tag{4.56}$$

V is the spatial volume occupied by the body. Another interpretation of the centre of mass is the point that is surrounded in all spatial directions by exactly the same amount of body mass:

$$0 = \int_V \rho(\boldsymbol{r})(\boldsymbol{r} - \boldsymbol{r}^c) \, dV. \tag{4.57}$$

The right-hand side expression in Eq. (4.56) is called the *zeroth order* moment of mass of the rigid body; the "zero" refers to the fact that the "moment arm" $\boldsymbol{r}$ appears in the integral to the zeroth power. The integral $\int \rho(\boldsymbol{r}) \boldsymbol{r} \, dV$ used in Eq. (4.55) is called the *first order* moment of mass; the subsequent Sections will also need the *second order* moment of mass $\int \rho(\boldsymbol{r}) \boldsymbol{r} \cdot \boldsymbol{r} \, dV$.

Newton's law for a rigid body

Applying the just-developed integration to Newton's law leads straightforwardly to the following reformulation: under the action of an external force $\boldsymbol{f}$, the centre of mass of a rigid body moves as if it were a point carrying all mass of the body:

$$\boldsymbol{f} = m \ddot{\boldsymbol{r}}^c = \frac{d\boldsymbol{p}^c}{dt}. \tag{4.58}$$

4.12.2 Dynamics of angular momentum

The previous paragraphs generalised Newton's equation of motion from a point mass to a rigid body with finite extension. A similar reasoning links the moment of force $\boldsymbol{m}$ to the time rate of the angular momentum of the rigid body. Indeed, first take the vector product of Eq. (4.50) with the "lever arm" $\boldsymbol{r}_i$ between the origin of the world reference frame and the point mass i in the rigid body:

$$\boldsymbol{r}_i \times \boldsymbol{f}_i^{\text{ext}} + \sum_j \boldsymbol{r}_i \times \boldsymbol{f}_{ji} = \boldsymbol{r}_i \times \frac{d\boldsymbol{p}_i}{dt}. \tag{4.59}$$

As for the angular momentum of a single point mass, the right-hand side can be rewritten as $d(\boldsymbol{r}_i \times \boldsymbol{p}_i)/dt$. Taking the sum over all points i gives

$$\sum_i \boldsymbol{r}_i \times \boldsymbol{f}_i^{\text{ext}} + \sum_{i,j} \boldsymbol{r}_i \times \boldsymbol{f}_{ji} = \sum_i \boldsymbol{r}_i \times \frac{d\boldsymbol{p}_i}{dt}. \tag{4.60}$$

The second sum is equal to

$$\sum_{i<j} (\boldsymbol{r}_i - \boldsymbol{r}_j) \times \boldsymbol{f}_{ji} = \sum_{i<j} \boldsymbol{r}_{ji} \times \boldsymbol{f}_{ji}. \tag{4.61}$$

Since the rigidity constraint forces $\boldsymbol{f}_{ji}$ work along the line between the points i and j, $\boldsymbol{r}_{ji}$ and $\boldsymbol{f}_{ji}$ are parallel, and their vector product vanishes. Hence, under the action of an external moment $\boldsymbol{m}$, a rigid body changes its angular momentum $\boldsymbol{l}$ according to the following law:

$$\boldsymbol{m} = \frac{d\boldsymbol{l}}{dt}. \tag{4.62}$$

4.12.3 Inertia tensor about centre of mass

The position vector $\boldsymbol{r}$ of each point p in a rigid body splits as follows:

$$\boldsymbol{r} = \boldsymbol{r}^c + \boldsymbol{r}^{c,p}, \tag{4.63}$$

where $\boldsymbol{r}^c$ is the position vector of the centre of mass with respect to the inertial world reference frame, and $\boldsymbol{r}^{c,p}$ is the vector from the centre of mass to the point in question. Together with Eqs (4.42) and (4.56), this gives the following expression for the angular momentum $\boldsymbol{l}$:

$$\begin{aligned}\boldsymbol{l} &= \int_V (\boldsymbol{r}^c + \boldsymbol{r}^{c,p}) \times \rho(\boldsymbol{r}) \left(\dot{\boldsymbol{r}}^c + \dot{\boldsymbol{r}}^{c,p}\right) dV \\ &= \boldsymbol{r}^c \times m\dot{\boldsymbol{r}}^c + \int_V \boldsymbol{r}^{c,p} \times \rho(\boldsymbol{r}) \frac{d}{dt}\boldsymbol{r}^{c,p}\, dV \\ &\quad + \boldsymbol{r}^c \times \frac{d}{dt}\int_V \rho(\boldsymbol{r})\boldsymbol{r}^{c,p}\, dV + \left(\int_V \rho(\boldsymbol{r})\boldsymbol{r}^{c,p}\, dV\right) \times \dot{\boldsymbol{r}}^{c,p},\end{aligned} \tag{4.64}$$

with $\dot{\boldsymbol{r}}^c$ the velocity of the centre of mass with respect to the inertial world reference frame. By the definitions of the linear momentum, Eq. (4.36), and the centre of mass, Eq. (4.55), the last two terms vanish, and one gets

$$\boldsymbol{l} = \boldsymbol{l}^l + \boldsymbol{l}^a = \boldsymbol{r}^c \times m\dot{\boldsymbol{r}}^c + \int_V \rho(\boldsymbol{r})\boldsymbol{r}^{c,p} \times \dot{\boldsymbol{r}}^{c,p}\, dV. \tag{4.65}$$

This means that the total angular momentum of a moving rigid body, as seen from an inertial reference frame, is the sum of (i) the angular momentum $\boldsymbol{l}^l$ of a point mass at the centre of mass carrying the mass of the complete rigid body, and (ii) the angular momentum $\boldsymbol{l}^a$ of the rigid body about the centre of mass. Using the vector identities depicted in Figure 4.5, $\boldsymbol{l}^a$ can be rewritten as

$$\begin{aligned}\boldsymbol{l}^a &= \int_V \rho(\boldsymbol{r})\boldsymbol{r}^{c,p} \times (\boldsymbol{\omega} \times \boldsymbol{r}^{c,p})\, dV \\ &= -\int_V \rho(\boldsymbol{r})\boldsymbol{r}^{c,p} \times (\boldsymbol{r}^{c,p} \times \boldsymbol{\omega})\, dV \\ &= -\left(\int_V \rho(\boldsymbol{r})[\boldsymbol{r}^{c,p}][\boldsymbol{r}^{c,p}]\, dV\right)\boldsymbol{\omega} \qquad (4.66) \\ &= \boldsymbol{I}\boldsymbol{\omega}. \qquad (4.67)\end{aligned}$$

$\boldsymbol{I}$ is the *(Cartesian) inertia tensor* (or (rotational) inertia matrix, [76]) of the rigid body, as expressed with respect to the centre of mass of the body. Equation (4.67) shows that the inertia tensor $\boldsymbol{I}$ can be interpreted as a *linear operator* that maps the angular velocity $\boldsymbol{\omega}$ of a moving rigid body onto the angular momentum $\boldsymbol{l}^a$ of the rigid body about its centre of mass.

As for the linear momentum of the rigid body, Eq. (4.36), one can (with a slight abuse of notation) rewrite its angular momentum as a *six-vector* screw $\mathbf{l}$:

$$\mathbf{l} = \begin{pmatrix} \mathbf{0} \\ \boldsymbol{I}\boldsymbol{\omega} \end{pmatrix}. \tag{4.68}$$

Euler's law at the centre of mass

The combination of (i) the angular momentum dynamics of a rigid body, Eq. (4.62), and (ii) the decoupling of the angular momentum about the centre of mass, gives Euler's law of angular motion dynamics: under the action of an external moment $\boldsymbol{m}$, a rigid body changes its angular motion *about the centre of mass* according to Euler's law (1750):

$$\boldsymbol{m} = \frac{d}{dt}(\boldsymbol{I}\boldsymbol{\omega}). \tag{4.69}$$

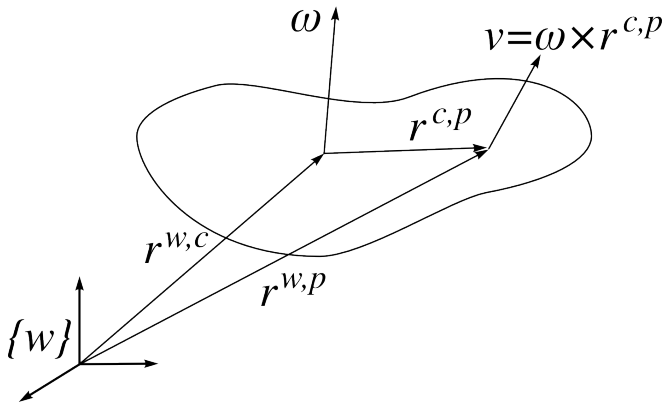

Figure 4.5: Relations between the vectors $\boldsymbol{r}$, $\dot{\boldsymbol{r}}$, and $\boldsymbol{\omega}$ of a rotating rigid body.

Inertia tensor in arbitrary frame

In Eq. (4.69), $\boldsymbol{I}$ is the inertia tensor about the centre of mass. Recall that the vector $\boldsymbol{r}^{c,p}$ in Eq. (4.66) denotes the position vector from the centre of mass to a point in the body. If, with respect to a reference frame $\{f\}$, this vector $\boldsymbol{r}^{c,p}$ has components x, y, and z, then the coordinate representation of the inertia tensor in this frame $\{f\}$ is

$$ {}_f\boldsymbol{I} = \int_V \rho({}_f\boldsymbol{r}) \begin{pmatrix} y^2+z^2 & -xy & -xz \\ -xy & x^2+z^2 & -yz \\ -xz & -yz & x^2+y^2 \end{pmatrix} dV. \tag{4.70} $$

Or, in more compact form:

$$ {}_f\boldsymbol{I} = {}_c\boldsymbol{I} - m\left[\boldsymbol{r}^{c,p}\right]\left[\boldsymbol{r}^{c,p}\right]. \tag{4.71} $$

$[\boldsymbol{r}^{c,p}]$ is the 3×3 vector product matrix corresponding to $\boldsymbol{r}^{c,p}$. So, it is clear that the inertia tensor is a *symmetric matrix*, irrespective of the reference frame $\{f\}$ in which it is calculated. The physical units of the elements in the inertia tensor are mass $\times$ (length)^2.

Reference frames for inertia tensor

The definition of the inertia tensor depends on the choice of *two* reference frames:

1. One frame is fixed to the rigid body, and has its origin at the centre of mass. The vector $\boldsymbol{r}^{c,p}$ in the integral in Eq. (4.70) runs from this origin to each point in the rigid body.

2. A second frame is the "world" reference frame in which the *coordinates* of this vector are expressed.

This means that Eq. (4.70) can be used to calculate the same inertia tensor (i.e., with the centre of mass as reference point) with respect to different world frames: just use the same formal expressions, but fill in the coordinates of the vector $\boldsymbol{r}^{c,p}$ with respect to the chosen world reference frame. However, if one chooses a reference point in these integrals that differs from the centre of mass, the resulting inertia tensor has to be calculated differently, Section 4.12.6.

4.12.4 Kinetic energy

The decoupling about the centre of mass (as found in the momentum equations of a rigid body) is also reflected in the kinetic energy of the moving body:

1. The kinetic energy T of a *single point mass* with mass m and velocity $\dot{\boldsymbol{r}}$ (with magnitude $|\dot{\boldsymbol{r}}| = v$) is defined as

$$ T = \frac{mv^2}{2} = \frac{m}{2}\dot{\boldsymbol{r}}\cdot\dot{\boldsymbol{r}}. \tag{4.72} $$

2. This generalises straightforwardly to the kinetic energy of a *rigid body*, by integration over the whole body:

$$\begin{aligned} T &= \frac{1}{2}\int_V \rho(\boldsymbol{r})\dot{\boldsymbol{r}}\cdot\dot{\boldsymbol{r}}\,dV \\ &= \frac{1}{2}\int_V \rho(\boldsymbol{r})(\dot{\boldsymbol{r}}^c+\boldsymbol{\omega}\times\boldsymbol{r}^{c,p})\cdot(\dot{\boldsymbol{r}}^c+\boldsymbol{\omega}\times\boldsymbol{r}^{c,p})\,dV. \end{aligned}$$

Here, $\boldsymbol{\omega}$ is the angular velocity of the rigid body, and $\dot{\boldsymbol{r}}^c$ is the instantaneous linear velocity of the centre of mass with respect to the inertial world reference frame. With v^c denoting the magnitude of the velocity three-vector $\dot{\boldsymbol{r}}^c$, the kinetic energy becomes

$$\begin{aligned} T &= \frac{1}{2}\int_V \rho(\boldsymbol{r})\left((v^c)^2+2\dot{\boldsymbol{r}}^c\cdot(\boldsymbol{\omega}\times\boldsymbol{r}^{c,p})+(\boldsymbol{\omega}\times\boldsymbol{r}^{c,p})\cdot(\boldsymbol{\omega}\times\boldsymbol{r}^{c,p})\right)\,dV \\ &= \frac{1}{2}\,m(v^c)^2+\dot{\boldsymbol{r}}^c\cdot\left(\boldsymbol{\omega}\times\int_V\rho(\boldsymbol{r})\boldsymbol{r}^{c,p}\,dV\right)+\frac{1}{2}\int_V\rho(\boldsymbol{r})(\boldsymbol{\omega}\times\boldsymbol{r}^{c,p})\cdot(\boldsymbol{\omega}\times\boldsymbol{r}^{c,p})\,dV. \end{aligned}$$

The definition of the centre of mass makes the middle term of this expression equal to zero, and the first term equal to the kinetic energy of a point mass carrying all the mass of the rigid body, and moving with the speed of the centre of mass of the body. The third term (with the superscript "c,p" omitted for brevity) can be developed as

$$\begin{aligned} \frac{1}{2}\int_V\rho(\boldsymbol{r})(\boldsymbol{\omega}\times\boldsymbol{r})\cdot(\boldsymbol{\omega}\times\boldsymbol{r})\,dV &= \frac{1}{2}\int_V\rho(\boldsymbol{r})(\boldsymbol{r}\times\boldsymbol{\omega})\cdot(\boldsymbol{r}\times\boldsymbol{\omega})\,dV \\ &= \frac{1}{2}\int_V\rho(\boldsymbol{r})\boldsymbol{\omega}^T[\boldsymbol{r}]^T[\boldsymbol{r}]\boldsymbol{\omega}\,dV \\ &= \frac{1}{2}\,\boldsymbol{\omega}^T\left(\int_V\rho(\boldsymbol{r})[\boldsymbol{r}]^T[\boldsymbol{r}]\,dV\right)\boldsymbol{\omega} \\ &= \frac{1}{2}\,\boldsymbol{\omega}^T\boldsymbol{I}\,\boldsymbol{\omega}, \end{aligned}$$

with the inertia tensor $\boldsymbol{I}$ as in Eq. (4.66). Hence, the total kinetic energy of the moving rigid body is

$$T=\frac{1}{2}m(v^c)^2+\frac{1}{2}\boldsymbol{\omega}^T\boldsymbol{I}\,\boldsymbol{\omega}. \tag{4.73}$$

Hence, this kinetic energy, like the angular momentum, also *decouples* in (i) the kinetic energy of the centre of mass, and (ii) the kinetic energy of the body's rotation about the centre of mass.

Centroidal reference frame. It is clear from Eq. (4.73) that the inertia tensor $\boldsymbol{I}$ is *positive-definite*, since the kinetic energy is always positive. Hence, $\boldsymbol{I}$ can be diagonalised, or, in other words, there exists an orthogonal reference frame with origin at the centre of mass, such that the inertia tensor is diagonal when expressed with respect to this so-called *centroidal reference frame*.

4.12.5 Potential energy in gravitational field

The potential energy V of a point mass m, at a position $\boldsymbol{r}$ with respect to a reference point, and due to the gravitational field of the earth, is

$$V=m\boldsymbol{g}\cdot\boldsymbol{r}, \tag{4.74}$$

where the three-dimensional vector $\boldsymbol{g}$ is the gravitational acceleration. The potential energy for a collection of point masses assembled in one single rigid body is then given by the integral over the total body:

$$V=\int_V\rho(\boldsymbol{r})\boldsymbol{g}\cdot\boldsymbol{r}\,dV=\boldsymbol{g}\cdot\int_V\rho(\boldsymbol{r})\boldsymbol{r}\,dV=m\,\boldsymbol{g}\cdot\boldsymbol{r}^c, \tag{4.75}$$

with m the total mass of the rigid body, and $\boldsymbol{r}^c$ the position vector of the centre of mass of the body. Hence, in contrast to the kinetic energy, the potential energy of a moving rigid body only depends on the *position* of the centre of mass, and not on the instantaneous velocity of the body, nor on the mass distribution over the body.

4.12.6 Mass matrix

The kinetic energy in Eq. (4.73) uses the three-vectors $\boldsymbol{\omega}$ and $\dot{\boldsymbol{r}}^c$ to represent the velocity of the rigid body. These two three-vectors together form a *pose twist* $\mathbf{t}$ of the body (Sect. 4.2.2):

$$\mathbf{t} = \begin{pmatrix} \boldsymbol{\omega} \\ \dot{\boldsymbol{r}}^c \end{pmatrix}. \tag{4.76}$$

A *pose twist* representation occurs here as the natural choice of velocity representation: the velocity of the moving rigid body is represented in an arbitrary inertial frame by (i) the translational velocity three-vector of a *velocity reference point that coincides with the centre of mass*, and (ii) the angular velocity three-vector of the body. With Eq. (4.76), the kinetic energy can be rewritten as

$$T = \frac{1}{2} m \left(\dot{\boldsymbol{r}}^c\right)^T \dot{\boldsymbol{r}}^c + \frac{1}{2} \boldsymbol{\omega}^T \boldsymbol{I} \boldsymbol{\omega} = \frac{1}{2} \mathbf{t}^T \begin{pmatrix} \boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & m\mathbf{1} \end{pmatrix} \mathbf{t} = \frac{1}{2} \mathbf{t}^T \boldsymbol{M} \mathbf{t}. \tag{4.77}$$

This generalises the well-known expression for the kinetic energy of a point mass m moving with velocity v: $T = mv^2/2$, so, the 6×6 matrix $\boldsymbol{M}$ is called the *(Cartesian) mass matrix* (or "spatial inertia matrix", [76], or also "generalised inertia matrix", [173]) of the rigid body with respect to the centre of mass. The different entries in the mass matrix have different physical units, but the kinetic energy is a positive *scalar*. As a 1×1 matrix, it is equal to its own transpose, so $\boldsymbol{M}$ is *positive-definite* (and invertible) and *symmetric*.

Equation (4.77) expresses the kinetic energy *with the centre of mass as velocity reference point*. Remember that screw and pose twists (Sect. 4.2.2) differ in the reference point that is chosen for the translational velocity three-vector in the twist: screw twists take the origin of the world reference frame as velocity reference point, pose twists take a point on the moving body as velocity reference point. However, if the world reference frame is coincident with the body-fixed reference frame whose origin serves as the velocity reference point for the pose twist representation, then pose twist and screw twist are identical. So, Eq. (4.77), expressed with respect to the body-fixed reference frame serving also as the world reference frame, yields

$$T = \frac{1}{2} {}_c\mathbf{t}^T \begin{pmatrix} {}_c\boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & m\mathbf{1} \end{pmatrix} {}_c\mathbf{t} = \frac{1}{2} {}_c\mathbf{t}^T {}_c\boldsymbol{M} \, {}_c\mathbf{t} \tag{4.78}$$

with the twist ${}_c\mathbf{t}$ either a pose twist or a screw twist, expressed in a frame with origin at the centre of mass of the body. It is straightforward to find how the mass matrix *transforms* its coordinate expression ${}_c\boldsymbol{M}$ (for *screw twists*!) in the centroidal frame $\{c\}$ to its coordinate expression ${}_f\boldsymbol{M}$ in an arbitrary frame $\{f\}$, from the observation that the kinetic energy should be the same in both representations, ${}_c\mathbf{t}^T \, {}_c\boldsymbol{M} \, {}_c\mathbf{t} = {}_f\mathbf{t}^T \, {}_f\boldsymbol{M} \, {}_f\mathbf{t}$, hence:

$${}_f\boldsymbol{M} = {}^f_c\boldsymbol{S}^T \, {}_c\boldsymbol{M} \, {}^f_c\boldsymbol{S}. \tag{4.79}$$

4.12.7 Euler's law of motion revisited

Euler's equation of motion is most compactly given by Eq. (4.69), but its most commonly known form is found from that equation by explictly developing the time derivative of $\boldsymbol{I}\boldsymbol{\omega}$. So, assume that the moment $\boldsymbol{m}$, the inertia matrix $\boldsymbol{I}$, and the angular velocity $\boldsymbol{\omega}$ of the rotating rigid body are all expressed with respect to an inertial world reference frame $\{w\}$: ${}_w\boldsymbol{m}$, ${}_w\boldsymbol{I}$, and ${}_w\boldsymbol{\omega}$. On the other hand, assume that the inertia tensor is known with respect to a body-fixed reference fame $\{b\}$ at the centre of mass of the body, and that the orientation between $\{b\}$ and $\{w\}$ is given by the rotation matrix ${}^b_w\boldsymbol{R}$. Due to the motion of the body, this rotation matrix varies over time. Hence

$$\begin{aligned}
{}_w\boldsymbol{m} &= \frac{d}{dt}({}_w\boldsymbol{I} \, {}_w\boldsymbol{\omega}), \\
&= \frac{d}{dt}({}^w_b\boldsymbol{R}^T \, {}_b\boldsymbol{I} \, {}^w_b\boldsymbol{R}) \, \boldsymbol{\omega}, \\
&= {}^w_b\dot{\boldsymbol{R}}^T \, {}_b\boldsymbol{I} \, {}^w_b\boldsymbol{R}\boldsymbol{\omega} + {}^w_b\boldsymbol{R}^T \, {}_b\boldsymbol{I} \, {}^w_b\dot{\boldsymbol{R}}\boldsymbol{\omega} + {}^w_b\boldsymbol{R}^T \, {}_b\boldsymbol{I} \, {}^w_b\boldsymbol{R}\dot{\boldsymbol{\omega}}, \\
&= {}^w_b\dot{\boldsymbol{R}}^T \, ({}^b_w\boldsymbol{R}^T \, {}_w\boldsymbol{I} \, {}^b_w\boldsymbol{R}) {}^w_b\boldsymbol{R} \, {}_w\boldsymbol{\omega} + {}^w_b\boldsymbol{R}^T ({}^b_w\boldsymbol{R}^T \, {}_w\boldsymbol{I} \, {}^b_w\boldsymbol{R}) {}^w_b\dot{\boldsymbol{R}} \, {}_w\boldsymbol{\omega} + {}_w\boldsymbol{I} \, {}_w\dot{\boldsymbol{\omega}},
\end{aligned}$$

and using Eqs (3.10) and (3.20):

$${}_w\boldsymbol{m} = [{}_w\boldsymbol{\omega}] \, {}_w\boldsymbol{I} \, {}_w\boldsymbol{\omega} - {}_w\boldsymbol{I} \, [{}_w\boldsymbol{\omega}] \, {}_w\boldsymbol{\omega} + {}_w\boldsymbol{I} \, {}_w\dot{\boldsymbol{\omega}}.$$

The second term on the right-hand side vanishes, and what remains is the well-known alternative form of Euler's equation:

$$ {}_w\boldsymbol{m} = {}_w\boldsymbol{I}\,{}_w\dot{\boldsymbol{\omega}} + [{}_w\boldsymbol{\omega}]\,{}_w\boldsymbol{I}\,{}_w\boldsymbol{\omega} = {}_w\boldsymbol{I}\,{}_w\dot{\boldsymbol{\omega}} + {}_w\boldsymbol{\omega} \times {}_w\boldsymbol{I}\,{}_w\boldsymbol{\omega}. \tag{4.80} $$

Since ${}_b\boldsymbol{m} = {}^w_b\boldsymbol{R}\,{}_w\boldsymbol{m}$, ${}_b\boldsymbol{\omega} = {}^w_b\boldsymbol{R}\,{}_w\boldsymbol{\omega}$, and ${}_b\boldsymbol{I} = {}^b_w\boldsymbol{R}^T\,{}_w\boldsymbol{I}\,{}^b_w\boldsymbol{R}$, Euler's equation in a body-fixed reference frame is

$$ {}_b\boldsymbol{m} = {}_b\boldsymbol{I}\,{}_b\dot{\boldsymbol{\omega}} + [{}_b\boldsymbol{\omega}]\,{}_b\boldsymbol{I}\,{}_b\boldsymbol{\omega}. \tag{4.81} $$

Note that the equations (4.80) and (4.81) dissect the right-hand side of $\boldsymbol{m} = d(\boldsymbol{I\omega})/dt$ in two parts:

1. $\boldsymbol{I}\dot{\boldsymbol{\omega}}$ is the rotational equivalent of the familiar form $m\dot{\boldsymbol{v}}$ of Newton's equation of motion.
2. $\boldsymbol{\omega} \times \boldsymbol{I\omega}$ vanishes if the body is not rotating, i.e., $\boldsymbol{\omega} = 0$, *or* if the mass distribution around the origin of the reference frame is *isotropic*. In this case, the inertia matrix is a multiple of the identity matrix and both terms in the vector cross product are parallel.

The second term arises because not all points in the rigid body move with the same speed if the body is rotating.

Practical form of Euler's equations

Although Eq. (4.69) is more compact, Eq. (4.81) is most often used, because all its components can be *measured* and ${}_b\boldsymbol{I}$ is a *constant*.

4.12.8 Newton-Euler equations for a single rigid body

This Section collects the results of all previous Sections into the general dynamic equations of a rigid body, expressed in a reference frame independent way. The first result is Eq. (4.86), which expresses the dynamics of a rigid body in a very concise form, which is formally completely similar to Newton's law for the dynamics of a point mass. However, this concise form is not practically useful, so it is transformed in a more elaborate equation, Eq. (4.89), (with, of course, the same physical interpretation!) that contains only terms that can be measured in a robot with traditional sensors.

The *total momentum* of the rigid body is the sum of its linear momentum $m\dot{\boldsymbol{r}}^c$, Eq. (4.36), and its angular momentum $\boldsymbol{I\omega}$ *about the centre of mass*, Eqs (4.65) and (4.67). The total momentum is an *invariant* property of the body's dynamics, i.e., independent of the chosen representation. Of course, it does depend on both the body's instantaneous velocity and its inertia. With respect to an arbitrary reference frame $\{f\}$, the total momentum can be represented by a six-vector $\boldsymbol{p}$:

$$ \boldsymbol{p} = \begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{r} \times \boldsymbol{d} + \boldsymbol{m} \end{pmatrix}, \quad \text{with} \quad \begin{cases} \boldsymbol{d} & = m\,{}_f\dot{\boldsymbol{r}}^{f,c}, \\ \boldsymbol{r} & = {}_f\boldsymbol{r}^{f,c}, \\ \boldsymbol{m} & = {}_f\boldsymbol{I}\,{}_f\boldsymbol{\omega}. \end{cases} \tag{4.82} $$

$\boldsymbol{p}$'s first three-vector is the linear momentum of the body, expressed in $\{f\}$; the second three-vector is the body's angular momentum *about the origin of* $\{f\}$. $\dot{\boldsymbol{r}}^{f,c}$ is the velocity of the centre of mass, as seen from the current reference frame $\{f\}$. This velocity is related to the *screw* twist $\mathbf{t} = (\boldsymbol{\omega}^T\ \boldsymbol{v}^T)^T$ of the rigid body in $\{f\}$ as follows (we omit the obvious superscripts and subscripts): $\boldsymbol{v} = \dot{\boldsymbol{r}} + \boldsymbol{r} \times \boldsymbol{\omega}$. Hence

$$ \boldsymbol{p} = \begin{pmatrix} -m[{}_f\boldsymbol{r}^{f,c}] & m\boldsymbol{I}_{3\times 3} \\ {}_f\boldsymbol{I} - m[{}_f\boldsymbol{r}^{f,c}]^2 & m[{}_f\boldsymbol{r}^{f,c}] \end{pmatrix} \begin{pmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{pmatrix} = (\boldsymbol{\Delta}\,\boldsymbol{M})\,\mathbf{t}, \tag{4.83} $$

$\boldsymbol{p}$ is a *screw*, since it transforms as (Sect. 2.7.3):

$$ {}_f\boldsymbol{p} = (\boldsymbol{\Delta}\,{}_f\boldsymbol{M})\,{}_f\mathbf{t} = \boldsymbol{\Delta}\left({}^f_i\boldsymbol{S}^T\,{}_i\boldsymbol{M}\,{}^f_i\boldsymbol{S}\right)({}^i_f\boldsymbol{S}_i\,\mathbf{t}) = \left(\boldsymbol{\Delta}\,{}^f_i\boldsymbol{S}^T\boldsymbol{\Delta}\right)(\boldsymbol{\Delta}\,{}_i\boldsymbol{M})\,{}_i\mathbf{t} = {}^i_f\boldsymbol{S}\,{}_i\boldsymbol{p}. \tag{4.84} $$

The reasoning above relies on (i) $\boldsymbol{\Delta}\,\boldsymbol{\Delta} = \mathbf{1}$, (ii) $\boldsymbol{\Delta}\,{}^f_i\boldsymbol{S}^T\boldsymbol{\Delta} = {}^i_f\boldsymbol{S}$, Eq. (2.29), and (iii) the transformation property of the mass matrix, Eq. (4.79). The equations above contain the $\boldsymbol{\Delta}$ operator, which has no physical meaning, but is only required to get the motion and force three-vectors in the *coordinate representations* of twists and momentum in a compatible order. The $\boldsymbol{\Delta}$ operator can be eliminated, but that introduces another notational "cost": the transformation of momentum, Eq. 4.84, changes, introducing an explicit $\boldsymbol{\Delta}$.

As remarked above, the dynamics of the rigid body decouple at the centre of mass:

$$ {}_c\mathsf{F} = \begin{pmatrix} {}_c\boldsymbol{f} \\ {}_c\boldsymbol{m} \end{pmatrix} = \begin{pmatrix} \frac{d}{dt}(m\,{}_c\dot{\boldsymbol{r}}) \\ \frac{d}{dt}({}_c\boldsymbol{I}\,{}_c\boldsymbol{\omega}) \end{pmatrix}, \tag{4.85} $$

or

$$ \mathsf{F} = \frac{d\,{}_c\boldsymbol{p}}{dt}. \tag{4.86} $$

This equation is the most compact way in which to express the full dynamics of an (unconstrained!) rigid body. However, no sensors exist that can measure the momentum (or its time derivative) directly, so this expression has only theoretical value: it shows that momentum is a basic physical variable, and it makes the formal resemblance with the dynamics of a point mass complete, Eq. (4.36).

The time derivative of the momentum screw $\boldsymbol{p}$ consists of two components: (i) a change due to a change in the instantaneous motion parameters $\boldsymbol{\omega}$ and $\boldsymbol{v}$, and (ii) a change due to the fact that the body moves to another place. The latter change of a screw is the *motor product* "$\times$" for screws, Sect. 4.10, Eq. (4.34). Hence, Eq. (4.86) results in an equation that is formally identical to Euler's equation of motion, Eq. (4.80):

$$ \mathsf{F} = (\boldsymbol{\Delta}\,\boldsymbol{M})\,\dot{\mathbf{t}} + \mathbf{t} \times (\boldsymbol{\Delta}\,\boldsymbol{M})\,\mathbf{t}. \tag{4.87} $$

Not surprisingly, this equation is called the *Newton-Euler equation of motion* for the rigid body. Since all entities used in the equations above are *screws*, the same formulas describe the dynamics of the rigid body with respect to any reference frame, after transformation with the appropriate screw transformation matrix. The Newton-Euler equation (4.87) has two components:

1. $(\boldsymbol{\Delta}\,\boldsymbol{M})\,\dot{\mathbf{t}}$: if the rigid body *is at rest*, this part of the wrench gives the body an acceleration according to a relationship that is formally identical to the acceleration for a point mass, i.e., Newton's equation of motion, Eq. (4.35).

2. $\mathbf{t} \times (\boldsymbol{\Delta}\,\boldsymbol{M})\,\mathbf{t}$: this part of the wrench on the rigid body is needed to keep the body in motion *without acceleration or deceleration*. This term is sometimes called the *bias force*, [74, 76], since it constitutes the difference with the more familiar Newton-like law of motion. That is, the *Coriolis and centrifugal forces*.

The derivations above show that the dynamics of a single rigid body maintain the following interesting properties from the dynamics of a point mass:

- The relationship between mass, force and acceleration is *linear*.
- The force has a *quadratic* dependency on the angular velocity.

The validity of the Newton-Euler equation can be intuitively corroborated by looking at the simple case of a rigid body rotating with constant speed about a fixed axis, such as, for example, a wooden horse on a merrry-go-round. In this case, the screw twist rate $\dot{\mathbf{t}}$ vanishes (because neither the screw axis, nor the magnitude of the velocity, changes) and the force needed to keep the body on its circular orbit is:

$$ \mathsf{F} = \mathbf{t} \times (\boldsymbol{\Delta}\,\boldsymbol{M})\,\mathbf{t}. \tag{4.88} $$

For a point with mass m, rotating at a distance r from the centre of rotation, and with an angular speed ω, this reduces to the well-known centrifugal force: $f = mr\omega^2$.

4.12.9 Practical Newton-Euler equations

This Section reformulates Eq. (4.87) into a form that contains only measurable or known components, and that uses notation that is more commonly used in the literature, and in segment-to-segment recursions such as in Sect. 5.4. Let $\boldsymbol{F} = (\boldsymbol{f}, \boldsymbol{m})$ denote the *(generalized) force* on the body, $\dot{\boldsymbol{X}} = (\boldsymbol{\omega}, \boldsymbol{v})$ the velocity, and $\ddot{\boldsymbol{X}} = (\dot{\boldsymbol{\omega}}, \dot{\boldsymbol{v}})$ the *acceleration*, then it is straightforward to derive the following expression for the Newton-Euler dynamics of a single, unconstrained rigid body:

$$ \boldsymbol{F} = \boldsymbol{\Delta M}\,\ddot{\boldsymbol{X}} + \boldsymbol{F}^b \qquad \text{with} \quad \boldsymbol{F}^b = \begin{pmatrix} \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} \\ \boldsymbol{\omega} \times (\boldsymbol{\omega} \times m\boldsymbol{r}^c) \end{pmatrix}. \tag{4.89} $$

$\boldsymbol{F}^b$ is the *bias force* already mentioned in Sect. 4.12.8, 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 (then $\boldsymbol{r}^c = \boldsymbol{0}$), *and* (ii) the body is spherical (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.

The mass matrix and its coordinate representations and transformations use the *geometrical parameters* of the mechanical robot structure, 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 $\boldsymbol{I}$. These parameters are in general only known *approximately*, such that they should be considered as *uncertain parameters* in the robot model.

4.12.10 Interpretation of mass matrix

The Cartesian mass matrix appears in two places: (i) the potential energy of a moving rigid body (Sect. 4.12.6), and (ii) the Newton-Euler equation of motion, Eq. (4.87). This yields the following interpretations:

1. $\boldsymbol{\Delta M}$ is a linear mapping from the vector space of *velocity* twists $\mathbf{t}$ to the *dual* vector space of momentum screws $\boldsymbol{p}$). This is very similar to the relationship between twists and wrenches. The action ("pairing") of a twist on a wrench is the *spatial scalar product*, and results in a number with the physical units of *power* (Sect. 2.6.3). Similarly, the pairing of a twist on a momentum results in a number with the physical units of *(kinetic) energy*.

2. $\boldsymbol{\Delta M}$ is a linear mapping from the vector space of *acceleration* twists $\dot{\mathbf{t}}$ to the *dual* vector space of wrenches $\boldsymbol{F}$).

3. $\boldsymbol{\Delta M}$ is a linear mapping from the vector space of twists $\mathbf{t}$ to the *dual* vector space of momentum screws $\boldsymbol{p}$). This is very

4. Since the physical interpretation of the mass matrix is that of an operator mapping (i) a twist into a momentum, Eq. (4.83), or (ii) a time derivative of a twist into a wrench, Eq. (4.87), the concepts of eigenvalue and eigenvector have no physical meaning. Those concepts only make sense for operators *within the same space*.

5. The ith *column* of $\boldsymbol{\Delta M}$ is the wrench needed to give the body a *unit acceleration* (from rest!) in the corresponding Cartesian direction.

6. The ith *element on the diagonal* of $\boldsymbol{M}$ is the kinetic energy generated by a unit velocity along the ith Cartesian direction.

4.13 Motion constraints

This Section explains how to deal with *physical* constraints, for example: a free-space motion specified in 6D while less than six joints are available to generate the motion, or motion with a six-jointed robot in contact with a stiff environment. The former gives rise to an optimization problem in Cartesian space (i.e., how *to approximate* the specified motion as good as possible); the latter gives rise to an optimization problem in joint space, similar to redundancy resolution, Sect. 5.9.1 (i.e., how to use all six joints to optimally execute a motion in the less than six-dimensional constrained Cartesian space). This text distinguishes between two complementary types of constraint *models*:

1. *Kinematic constraints* (also called *geometric* constraints, or *hard* constraints, or *algebraic* constraints): the instantaneous *velocities* that the robot can execute form (part of) a vector space with dimension lower than six. Mathematically, a kinematic constraint is *represented* by an *equality constraint* on the motion parameters; these are the velocities, in this case, but equality constraints on positions or accelerations are also quite common. The mathematical formulation as an equality constraint implicitly implies that the

constraint is *infinitely rigid*, i.e., it can take on any *constraint force* without the constraint equation being violated.

A constraint that can be expressed as an equality constraint on *positions* is called *holonomic constraint*; one that cannot be expressed at position level, but only as an equality constraint on *velocities* is called *non-holonomic*. The space of possible twists becomes lower-dimensional, since the motion of the rigid body is constrained in certain directions, not necessarily by contact with another rigid body, but by certain physical relationships that hold between some physical properties of the moving rigid body. For example: the velocities of all tyres in a car are related to each other (at least, when the car is not slipping); a moving rigid body obeys the physical law of conservation of angular momentum.

The space of remaining degrees-of-freedom 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. 2.6.2.

2. *Dynamic constraints* (or *impedance* constraints, or *soft* constraints):

   - *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.

A mechanical limit of a revolute joint is a simple example of a kinematic (*inequality*) 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. 7.8. 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 segment).

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.

4.13.1 Impedance—Admittance

Stiffness, damping and mass/inertia are the dynamical relationships between the zeroth, first and second derivative of position on the one hand, and force on the other hand. All real-world objects have their own specific stiffness, damping, and inertia; together, they form the so-called *impedance* of the object. The inverse of an impedance is called an *admittance*, [112].

The *first-order* properties of physical impedances can be represented by *matrices*: a *stiffness matrix*, (or *elasticity matrix*); a *damping matrix*, and an *inertia matrix*, or *mass matrix*, which was already extensively discussed in Sect. 4.12.6. The inverses of these mappings are called *compliance*, *accommodation*, and *mobility*, respectively. For example, the stiffness matrix $\boldsymbol{K}$ works on a displacement twist $\mathbf{t}_\Delta$ to produce a wrench $\mathbf{F}$: $\mathbf{F} = \boldsymbol{\Delta C}\,\mathbf{t}_\Delta$. The *coordinate transformations* of impedance matrices under a change of reference frame are easily derived from the transformation of the twists and wrenches they act on, similar to what was done for the mass matrix, Eq. (4.79); for example, the stiffness matrix $\boldsymbol{K}$:

$$ {}_f\boldsymbol{K} = {}^f_i\boldsymbol{S}^T {}_i\boldsymbol{K}\, {}^f_i\boldsymbol{S}. \tag{4.90} $$

4.13.2 Holonomic and non-holonomic constraints

Often, constraints are expressed at the *velocity* level, but in some cases they can also be expressed as ("integrated" into) a constraint on *positions*. If the velocity constraint is *integrable* into a position constraint, the constraint is called *holonomic*. This name comes from the Greek word *holos*, which means "whole", "integer." A (non)holonomic constraint on *velocities* can (not) be integrated to give a constraint on *positions*. An example of a non-holonomic constraint is the motion constraint imposed by skates on a skater, or by wheels on a car: the skater and car can only move in the gliding/rolling directions of the skate and the wheel, and not orthogonal to it; this kind of instantaneous motion constraint can not be expressed as a constraint on the position (and orientation) of the skater or the car, since they can reach any position or orientation they like. An example of a holonomic constraint is the *vertex-face* contact of Fig. 5.10: the position of the vertex must lie in the surface, and this can be "derived" into the constraint that the velocity of the point should be tangent to the surface.

Assume that the m non-holonomic constraints are of the following "Pfaffian" form, [173, 223, 230], named after the German mathematician Johann Friedrich Pfaff (1765–1825):

$$\boldsymbol{A}^T(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{0}, \quad \text{with} \quad \boldsymbol{A}(\boldsymbol{q}) \in \boldsymbol{R}^{n\times m}. \tag{4.91}$$

These constraints eliminate one or more instantaneous velocity degrees of freedom, or, in other words, they model possible constraint forces that force the system to move in accordance with the constraints. These forces are *ideal* constraint forces, if they perform *no work* on the system, i.e., the constraint forces $\boldsymbol{\tau}^c$ lie in the null space of the constraint matrix $\boldsymbol{A}(\boldsymbol{q})$:

$$\boldsymbol{A}^T(\boldsymbol{q})\boldsymbol{\tau}^c = 0. \tag{4.92}$$

4.13.3 Principles of d'Alembert and Gauss

The fact that the constraint forces do no work is often called *d'Alembert's principle*, [173], named after the French mathematician Jean Le Rond d'Alembert (1717–1783).

(TODO: provide more detailed description.)

Chapter 5

Kinematic chains

A *kinematic chain* is the mechanical *interconnection* of a set of rigid bodies (called *segments*, or *links*) by means of *joints*, e.g., Fig. 5.1. The main mechanical purpose of a kinematic chain is *to move* its "end-effector frames" (tools, sensors, targets,...) in the three-dimensional Cartesian space as part of a certain task. The kinematic chain *transforms* the motion of the motors (or muscles) at the *joints* to the Cartesian motion of the end-effector frames. This Chapter discusses only the *instantaneous* properties of these transformations: *position* ("*displacement*"), *velocity*, and *acceleration*. Chapter 6 deals with the (instantaneous) *dynamic* properties, the relationships between *acceleration*, *force* and *inertia*. The *non-instantaneous motion* properties of rigid bodies and kinematic chains are the subject of Chap. **??**. Chapters 7, 8 and 10 present the particular kinematic and dynamic properties of serial, parallel and mobile chains with *specific* kinematic structures.

Each joint imposes some *motion constraints* on the segments it interconnects. Most joints in robots are *revolute*, a few are *translational* (or "*linear*") joints. Both impose five constraints on the motion freedom of the segments they connect, or, equivalently, they allow one single motion degree of freedom. In humans, the shoulder and pelvis joints are two of the many joints that allow three degrees of (rotational) motion freedom. (In fact, the shoulder joint strictly speaking also allows some small translational motions.)

So, the most important aspect of this Chapter is the *transformation* of kinematic properties between "joint space" and "Cartesian space". For example, what velocities should one give to the *motors* of a kinematic chain in order for (one or more of) its *end-effectors* to move with a specified instantaneous spatial velocity; this is the *inverse kinematics* of the chain. Or, conversely, what spatial velocity results from applying given velocities at the motors; this is the *forward kinematics*.

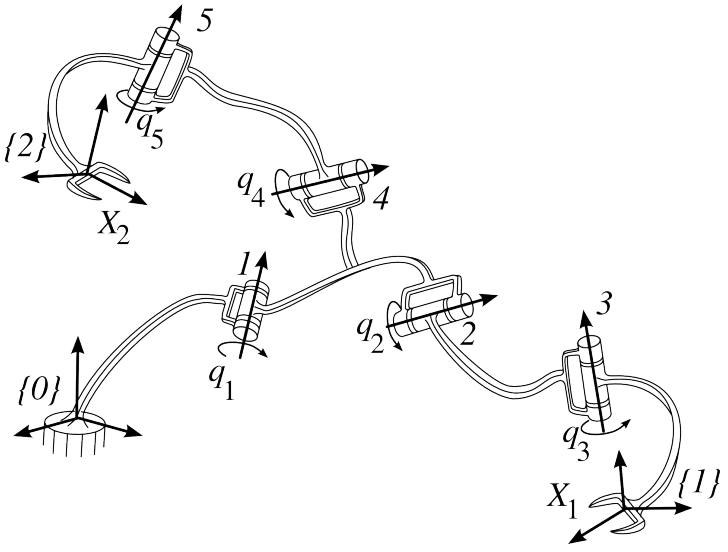

Figure 5.1: Kinematic chain, with the topological structure of a tree. It has one base (with reference frame {0}), two end-effectors (with reference frames {1} and {2}, and coordinate vectors $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$), and five revolute joints (with joint positions $q_1, \ldots, q_5$).

The mathematical relationship $\boldsymbol{f}$ between joint positions $\boldsymbol{q} = (q_1, \ldots, q_n)$ and end-effector poses $\boldsymbol{X} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_m)$ is, in general, a nonlinear function of the joint positions, and can be expressed in *implicit* form,

$$\boldsymbol{f}(\boldsymbol{X}, \boldsymbol{q}) = 0, \quad \text{(position closure)}, \tag{5.1}$$

or (two) *explicit* ("input-output") forms:

$$\begin{cases} \boldsymbol{X} & = \boldsymbol{g}(\boldsymbol{q}), \quad \text{(forward kinematics)}, \\ \boldsymbol{q} & = \boldsymbol{h}(\boldsymbol{X}), \quad \text{(inverse kinematics)}. \end{cases} \tag{5.2}$$

The geometric properties of the kinematic chain (i.e., dimensions of the segments, orientations of the joint axes, ...) are parameters in these relationships. The implicit form of the relationships is the most general, since kinematic chains exist for which the explicit form does not work well, in the sence that the output for a given input is not well-defined. For example, the fully parallel robots of Chap. 8 have multiple possible end-effector poses for the same set of (actuated) joint positions, and, due to the loops in the kinematic chain, not all sets of (actuated and non-actuated) joint positions correspond to mechanically feasible configurations of the kinematic chain. Similarly, most serial chains have multiple joint configurations that give the same end-effector pose.

The description above, about the relationships between joint space variables $\boldsymbol{q}$ and end-effector space variables $\boldsymbol{X}$, can be generalized to any physical variable that moves together with the kinematic chain. In particular, similar relationships hold for the motions of the ("end effector" frames attached to the) *passive joints* $\boldsymbol{q}^p$ (i.e., the ones *without* a motor attached to them to generate motion between the segments connected at the joint) and the (frames attached to the) *active joints* $\boldsymbol{q}^a$ in graph chains with closed loops, Sect. 5.5.4, [58]. (Passive joints are sometimes also called *non-actuated joints*.)

The discussion in this book allows multiple end-effectors on one kinematic chain, even on a serial chain: one can put multiple tools or sensors at different locations on the robot, and the robot programmer is interested to make each of these move in a specified way. Of course, using multiple end-effectors makes the kinematics problems harder to solve, and in some cases even no feasible solution exists, since the chain has too many mechanical constraints to satisfy the desired motion requirements of the programmer.

Most kinematic chains used in robotic devices have one-dimensional joints, that drive the connected segments in straightforward ways, and for which the mechanical position and force limits are straightforward to specify and to control. However, natural kinematic chains (such as the human *musculo-skeletal system*) often present more challenges to the mathematical representation of the "joints": many articulations in the body are not at all ideal revolute or linear joints; many muscles influence more than one joint, and apply forces on various and distributed regions on several segments. The current state of the art in the description and analysis of natural kinematic chains limits itself to "robotic" approximations of the reality.

5.1 Reachable and dextrous workspace

The *reachable workspace* of (one of) the kinematic chain's end-effector is the manifold of reachable frames for that end-effector, irrespective of how the chain must move to reach those frames. The *dextrous workspace* is the *subset* of the reachable workspace where the chain can generate velocities for that end-effector in all directions, i.e., it can translate the end-effector with three degrees of translation freedom, and rotate it with three degrees of rotation freedom, [138, 194].

The relationships between joint space and Cartesian space coordinates of the object held by the kinematic chain are in general *multiple-valued*: the same pose in the reachable workspace can be reached by the chain in different ways, each with a different set of joint coordinates. Hence, the reachable workspace of the chain is divided in *configurations* (also called *assembly modes*), in which the kinematic relationships are *locally* one-to-one. More details follow in the discussions on *inverse position kinematics* in later Sections.

5.2 Chebyshev-Kutzbach-Grübler mobility criterion

The *Industrial Revolution* of the 19th century gave rise to intensive research on *mechanisms* (also called "parallellograms", or "linkages"), i.e., kinematic chains were developed to transform the motion of the steam engine piston (the "joint" space) to the "end-effector" space of the steam-powered task. One focus of attention was the search for a *topological* criterion to quickly find *mobility* of kinematic chains with many segments and joints, that

is, the number of degrees of freedom that a mechanism allows, [5, 58, 83]. "Topological" means that the *geometry* of segments and joints are not taken into account, but that one only counts the *number* of segments and joints. The most often cited such criterion now carries the name of Chebyshev, Grübler and Kutzbach, and looks as follows:

$$m = 6(n - j - 1) + \sum_{n=1}^{j} f_i, \tag{5.3}$$

with m the mobility, n the number of segments (including the (unique) "ground" segment), j the total number of joints, regardless of their connectivity or degrees-of-freedom, and $\sum_{n=1}^{j} f_i$ the sum of all the degrees-of-freedom of all individual joints. (Various extensions have been described, see e.g. [83] for an overview.) Applying this formula to the example of a serial robot with six joints gives: $m = 6(7 - 6 - 1) + \sum_{i=1}^{6} 1 = 6$. Applying it to the example of the parallel robot in Fig. 5.9 gives: $n = 6$ legs $\times$ 5 segments in each leg (not counting the base and the end-effector) +2 (the base and the end-effector, which are shared by all legs); $j = 6 \times 6$ legs each with six joints); and $f_i = 1$ (each joint has one degree-of-freedom), so, in total $m = 6$ also.

The names of Pafnuty Lvovich Chebyshev (1821–1894), Martin Fürchtegott Grübler (1851–1935), and K. Kutzbach are connected to this criterion for the following reasons, [83]:

- *Chebyshev* published many articles on polynomial and approximation theory, which he (partially) developed to study the mobility of mechanisms. For example, he was searching—as many others in this time—for a mechanism with only revolute joints that could approximate a linear motion as good as possible. His most cited publication in this area is "Théorie des mécanismes connus sous le nom de parallélogrammes", [42], in which he laid the foundations of what would later become known as *Chebyshev polynomials*, and of which the above-mentioned problem was just one of the many possible applications. However, the written report of his lecture about this topic before the Sint Petersburg Academy is incomplete, and never explicitly mentions Eq. (5.3). The formula does appear in a later work, [43].

  [42] does *not* contain the CGK formula explicitly, but it might have ben discussed during the oral presentation that Chebyshev held at the Academy in Sint Petersburg, on January 28th, 1853, because:

  - the article on his lecture (see [42] for a translation into French from the original Russian) ends with a mention of the application of the theory of zeros of polynomials (which was the main topic of that presentation) to mechanisms
  - the "Rapport du professeur extraordinaire de l'université de St.Pétersbourg Tchebuchef sur son voyage à l'étranger" (volume II of Oeuvres de L. V. Chebyshev, pp. VII–XVIII) mentions the fact (p. XI) that "Comme j'avais très-peu de temps à ma disposition pour traiter un sujet de si grande étendue, je n'ai achevé que la première partie de mon Mémoire, qui a été présenté à l'Académie Impériale des Sciences." [43] contains the formula

    $$f = 3m - 2(n + v), \tag{5.4}$$

    with n = 1DOF revolute joints, m = segments, v = fixed points, and f = total degrees of freedom.

- *Grübler* published, in 1883, the first part of a two-part article, [92, 93], "Allgemeine Eigenschaften der Zwangläufigen Ebenen Kinematische Kette" ("General Characteristics of the Constrained Mobility of Planar Kinematic Chains"), in which he presents a first version of the mobility formula for the special case of planar chains. He summarized his work in his "Lehrbuch der technischen Mechanik" ("Textbook of Technical Mechanics") in 1921, [94].

- *Kutzbach* followed up on Grübler's work, [139], by reformulating his equation into the modern form of Eq. (5.3).

Many mechanisms exist that do not satisfy this topological criterion: they seem to have more degrees of freedom than straightforward application of the Chebyshev-Grübler-Kutzbach formula suggests, because their geometry is special. These mechanisms are sometimes called *overconstrained*, or *architecturally singular*. Mathematically speaking, this means that their Jacobians are *always* singular (Sect. 5.8), independent of the chain's joint configuration.

5.3 Segment frame conventions

The previous Sections describe *physical* properties of kinematic chains. But, in practice, all robot controllers must work with *coordinate representations* of kinematic chains and their properties, in order *to compute* the kinematic properties. This will be the topic of this and following Sections.

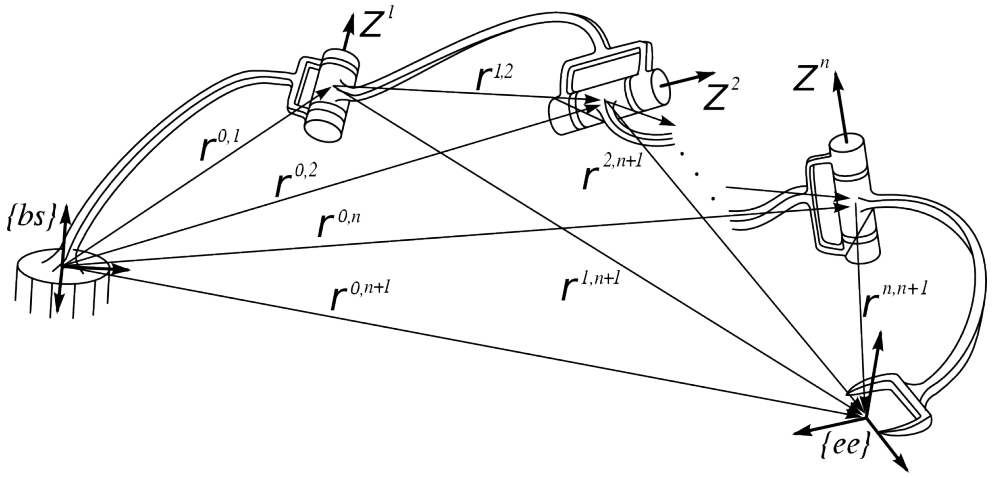

Figure 5.2: Reference frame and reference point model for a generic serial chain.

Coordinate representations for the kinematic models of robotic devices must allow to represent the relative pose, velocity and acceleration of two neighbouring segments, as a function of the position, velocity and acceleration of the joint connecting both segments. This Chapter assumes that the interconnections between segments and joints in a kinematic chain are all *ideal*: a joint is a perfect, geometric *motion constraint* for the rigid bodies that it connects: most joints allow one *degree of freedom* ("DOF") for the relative motion of both connected bodies, which means that it constrains the five other degrees of motion freedom. *Coordinate* representations of kinematic chains become simpler when *reference frames* are appropriately chosen, *and* simple conventions determine the choice of those frames. This Section presents some of the most common choices, with a discussion on the advantages and disadvantages of each.

Most often, one places a reference frame with its origin in the (ideal) joint axis, and with the unit vector of the Z axis along the motion degree of freedom that the joint allows. This choice introduces a lot of zeros in the coordinate representations, and the kinematic model is purely geometrical, as in Fig. 5.2 for the case of a *serial* kinematic chain. The "base" frame $\{bs\}$ gets the index "0", and, for a chain with n joints, the *end-effector* frame $\{ee\}$ has index "$n+1$." The direction vector $\boldsymbol{Z}^i$ represents the positive direction of the ith joint axis. The position vector $\boldsymbol{r}^{i,j}$ connects the origin of segment frame $\{i\}$ to the origin of segment frame $\{j\}$.

The segment closest to the base is sometimes called the *proximal link*; the link to which the end-effector is rigidly connected is the *distal* link. "Proximal" and "distal", as well as "base" and "end effector", tend to loose their meaning in non-serial chains or chains without a physically unique "base". Proximal and distal are being used in the case of humanoid skeletons too—these terms historically come from human anatomy!—even if those have a tree-like structure: the "base" of the anatomic nomenclature has been chosen to be the torso of the human, so the elbow joints are more distal than the shoulder joints, and the knee joint is more proximal than the ankle joint. Anyway, all above-mentioned segment frame conventions are used for non-serial structures too, and the only "property" that one looses in those cases is the fact that there exists no *natural* index ordening for all segments, as is the case in serial chains. The following Sections give some of the more common frame conventions.

5.3.1 Denavit-Hartenberg segment frame convention

Section 2.5.3 already introduced the Denavit-Hartenberg cooordinates of a *line*; this Section explains how to extend that line representation into a representation for the relative position and orientation of two *frames* on two chain segments connected by a joint. Since joint axes are, ideally, directed *lines* (at least for joints with a single degree of motion freedom), their representation needs (minimally) four parameters, and Fig. 5.3 shows these four parameters in the case of the Denavit-Hartenberg ("DH") convention, [65, 103]: $d^i, \alpha^{i-1}, \theta^i$ and h^{i-1} represent the joint axis Z^i of the ith joint (and the arbitrarily, but appropriately chosen frame axes X^i and Y^i

of the ith segment, with respect to the "proximal" (i.e., $i-1$) segment frame $\{X^i, Y^i, Z^i\}$, as a function of the $i-1$st joint motion.

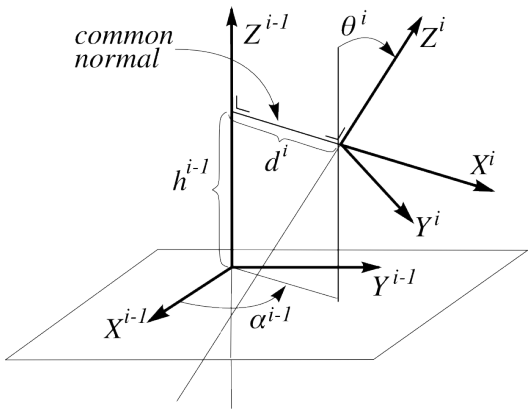

Figure 5.3: Frame definition according to one of the Denavit-Hartenberg conventions.

The relative pose of $\{i\}$ with respect to $\{i-1\}$ is defined as follows:

1. The joint axis is the Z^{i-1} axis. This means that the joint coordinate q^{i-1} is equal to the angle α^{i-1} if joint $i-1$ is revolute, and equal to the distance h^{i-1} if the joint is prismatic. The positive sense of the axis corresponds to the (arbitrarily chosen) 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.)

Not all references in the literature or not all software projects use the *same* 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. In addition, all DH conventions share some common problems:

- if the two lines are parallel, the DH convention before has a **singularity** here: a small change in the spatial positions of the lines can cause a very large change in the DH coordinate representation of their relative position. All other DH conventions will have similar coordinate singularities.

- 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.

- X^i is chosen to indicate a *fixed* direction with respect to the part of the segment that is fixed to the *proximal* joint $i-1$. That means that the joint value q^i of the joint that connects this segment to a distal segment 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.

The homogeneous transformation matrix ${}^{i}_{i-1}\boldsymbol{T}$ representing the relative pose of two subsequent segment frames $\{i-1\}$ and $\{i\}$ is straightforwardly derived from the DH convention:

$$
\begin{aligned}
{}^{i}_{i-1}\boldsymbol{T} &= \boldsymbol{R}(Z,\alpha^{i-1})\,\boldsymbol{Tr}(Z,h^{i-1})\,\boldsymbol{Tr}(X,d^{i})\,\boldsymbol{R}(X,\theta^{i}) \\
&= \begin{pmatrix} \cos(\alpha)^{i-1} & -\sin(\alpha)^{i-1} & 0 & 0 \\ \sin(\alpha)^{i-1} & \cos(\alpha)^{i-1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & h^{i-1} \\ 0 & 0 & 0 & 1 \end{pmatrix}
\begin{pmatrix} 1 & 0 & 0 & d^{i} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta)^{i} & -\sin(\theta)^{i} & 0 \\ 0 & \sin(\theta)^{i} & \cos(\theta)^{i} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},
\end{aligned}
\tag{5.5}
$$

or

$$
{}^{i}_{i-1}\boldsymbol{T} = \begin{pmatrix} \cos(\alpha)^{i-1} & -\sin(\alpha)^{i-1}\cos(\theta)^{i} & \sin(\alpha)^{i-1}\sin(\theta)^{i} & d^{i}\cos(\alpha)^{i-1} \\ \sin(\alpha)^{i-1} & \cos(\alpha)^{i-1}\cos(\theta)^{i} & -\cos(\alpha)^{i-1}\sin(\theta)^{i} & 0 \\ 0 & \sin(\theta)^{i} & \cos(\theta)^{i} & h^{i-1} \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{5.6}
$$

"$\boldsymbol{R}(Z,\alpha)$" represents the homogeneous transformation matrix that corresponds to the rotation about the Z-axis, over an angle α; "$\boldsymbol{Tr}(Z,h)$" represents the homogeneous transformation matrix that corresponds to the translation along Z over a distance h, etc.

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, since the *four* DH parameters cannot represent all *six* degrees of freedom that exist when positioning two arbitrary reference frames with respect to each other.

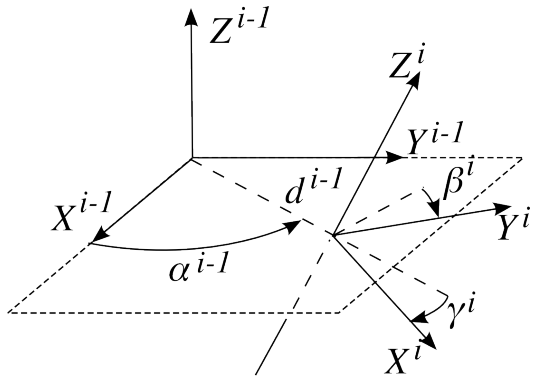

Figure 5.4: Frame definition in the Hayati-Roberts convention. (The Euler angles β^i and γ^i are not drawn completely accurately, in order not to clutter the figure with too many auxiliary lines.)

5.3.2 Hayati-Roberts segment frame convention

This section extends the Hayati-Roberts (HR) line convention (Sect. 2.5.4) to represent subsequent segment frames. It avoids the coordinate singularity in the DH convention for the case of (nearly) parallel lines. Similarly to the DH case, there is no unique HR convention, and Figure 5.4 shows only one possible definition. 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. The following four translations and rotations map the frame on the first joint axis onto the frame on the second joint axis:

$$
{}^{i}_{i-1}\boldsymbol{T} = \boldsymbol{R}(Z,\alpha^{i-1})\,\boldsymbol{Tr}(X,d^{i-1})\,\boldsymbol{R}(Y,\gamma^{i})\boldsymbol{R}(X,\beta^{i}). \tag{5.7}
$$

The "common normal" is not used in the HR convention (as it is the cause of the representation singularity in the DH convention), but 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), [19, 231].

5.3.3 Parametrically continuous segment frame convention

The Denavit-Hartenberg and Hayati-Roberts segment frame conventions result in a homogeneous transformation matrix that is a function of *four* geometric parameters. This means that these conventions can never represent an *arbitrary* pose transformation, since that requires six parameters. The missing "degrees of freedom" are (i) the position of the origin of the link frame along the joint axis (it *must* lie on the common normal), and (ii) the direction of the X-axis (it also *must* lie along the common normal). In addition, only for *modeling* purposes, four parameters are sufficient to represent all possible segment frames mathematically, because one can *choose* some parameters in these frames. However, in practice one often wants to make *smooth* adaptations to that mathematical model, because the mechanical construction of the robot or machine tool does not correspond completely to the model because of construction tolerances, wear, etc. Therefore, many robots are *calibrated*, that is, the "exact" relative pose of joints and segments are measured by a high-accuracy external measurement device. The consequence for the above-mentioned mathematical segment frame conventions is that the calibrated model could require more than four parameters, or cannot be represented by a "small" error on the four parameters used in the convention.

One of the answers to this calibration problem is the so-called *PC* segment frame convention, which is meant to be a *P*arametrically *C*ontinuous alternative to the DH convention. (It is part of a more elaborate convention (called *CPC convention*) designed by Zhuang *et al.*, [293].) The PC convention represents a line by:

1. Two direction cosines, b_x^i and b_y^i, being the X and Y projections of Z^i on the $X^{i-1}Y^{i-1}$ plane.

2. Two distance parameters, l_x^i and l_y^i, being the X and Y coordinates of the intersection of Z^i with the plane through the origin and orthogonal to Z^i.

The homogeneous transformation matrix for proximal-distal *frame* definitions, and corresponding to a set of PC parameters $(b_x^i, b_y^i, l_x^i, l_y^i)$, follows straightforwardly from the definition of these parameters:

$$ {}_{i-1}^{i}\boldsymbol{T} = \begin{pmatrix} 1-\dfrac{(b_x^i)^2}{1+b_z^i} & -\dfrac{b_x^i b_y^i}{1+b_z^i} & b_x^i & l_x^i \\ -\dfrac{b_x^i b_y^i}{1+b_z^i} & 1-\dfrac{(b_y^i)^2}{1+b_z^i} & b_y^i & l_y^i \\ -b_x^i & -b_y^i & b_z^i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad b_z^i = \sqrt{1-(b_x^i)^2-(b_y^i)^2}. \tag{5.8} $$

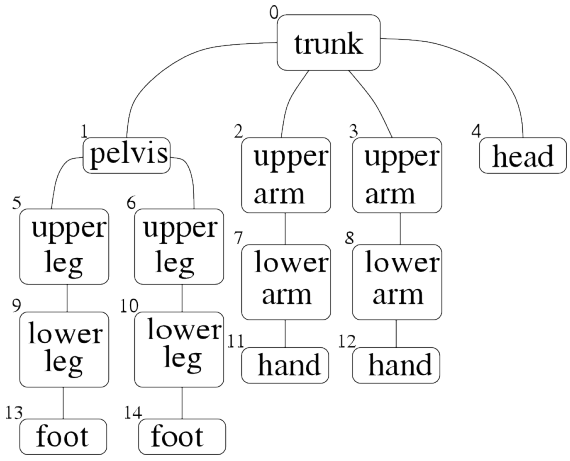

Figure 5.5: Tree topology of a simplified humanoid robot. The nodes are rigid body segments, the edges are joints. The choice of the trunk as segment "0" is rather common, since it corresponds to the anatomical definitions of *proximal* and *distal*: it allows to give numbers to all segments in such a way that a distal segment always has a higher number than the proximal segment it is connected to.

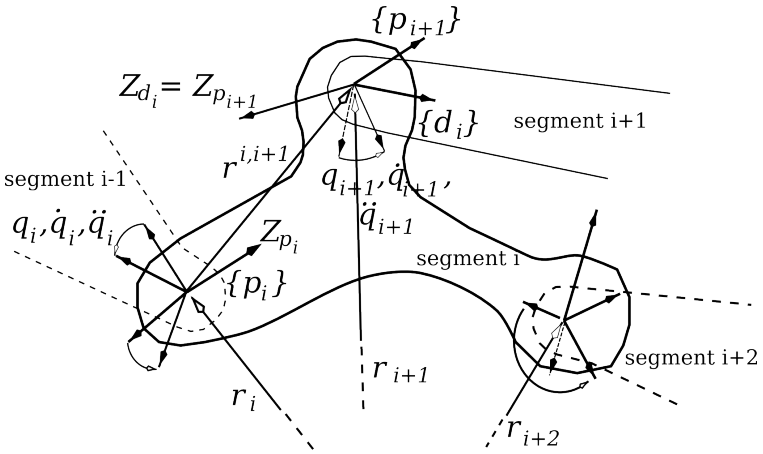

Figure 5.6: Reference frames and notation for segments in a tree-structured kinematic chain. "p_i" is the "proximal" frame of segment i, and "d_i" is the segment's distal frame. The frames' $\boldsymbol{Z}$ axes lie along the (single degree of freedom) axes of the joints connected to the segment. The convention in this book is that joint i "belongs" to the proximal frame $\{p_i\}$ of segment i.

5.3.4 Tree chain conventions

Tree structures are only slightly more complex than serial structures: every tree consists of serial sub-structures, the concepts of *proximal* and *distal* segments remain unchanged, with some segments having more than one distal segment. In a tree, there is *only one serial subchain* that connects any two segments. Figure 5.5 shows an example of a tree structure, namely a (highly simplified) version of the skeleton of a human, humanoid or walking robot. For each of the possibly multiple distal joints connected to a single segment, the serial segment conventions can be used without change. The added complexity of a tree structure with respect to a serial structure is only due to the bookkeeping of which segments are connected by which joints: trees have simple *serializations*, such as *breadth first* or *depth first* recursions through the tree. Figure 5.6 shows the notational conventions adopted in this text:

- Joint $i+1$ connects segments i and $i+1$, between the distal frame $\{d_i\}$ of segment i and the proximal frame $\{p_{i+1}\}$ of segment $i+1$.
- These frames $\{d_i\}$ and $\{p_{i+1}\}$ coincide when joint $i+1$ is in its *zero position*.
- For segments at which branching takes place, one segment i has multiple distal segments, but for notational simplicity the text only talks about distal segment $i+1$, since the mathematical formulations are always completely similar for all distal segments.

5.3.5 ISB convention for human skeleton chain

The *International Society of Biomechanics* (ISB) has standardized the frame definitions on the human skeleton, [284, 285], with most emphasis on the legs and the arms. The ISB convention uses only revolute joints, so it deviates significantly from reality in "joints" such as the shoulder and the wrist.

(TODO: explain ISB convention!)

5.3.6 X3D H-Anim convention for human skeleton chain

The international standard *X3D* (ISO/IEC 19774:2005, *Information technology – Computer graphics and image processing – Humanoid animation*) presents frame definitions on the human skeleton.

(TODO: explain H-Anim convention!)

5.3.7 Graph chain conventions

Kinematic chains with the topological structure of a *graph* are built from serial sub-structures, in a similar way as tree chains, but this time the serial sub-structures are connected in such a way that one or more *(kinematic) loops*

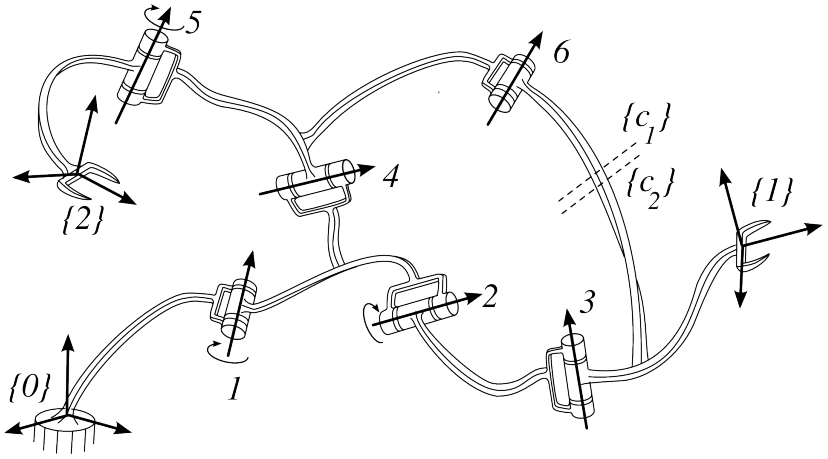

Figure 5.7: Graph chain, with one loop, over joints 2, 3, 4 and 6. (Not all joints must be actuated for this chain to be able to move deterministically; in this figure, 1, 2 and 5 are actuated.) Most kinematic algorithms cut the loop, indicated by the two dashed lines, resulting in a tree chain with two extra virtual "end-effector frames" $\{c_1\}$ and $\{c_2\}$.

exist, Figs 5.7 and 5.8. That means that some segments are connected to each other *via more than one serial sub-chain*. It also means that the segment frame conventions of serial chains can be applied to graph chains too, with some extra bookkeeping required to indicate between which segments the loops are closed. Deciding where to "cut a loop" is, in general, completely arbitrary, but it can have important consequences on the numerical computations.

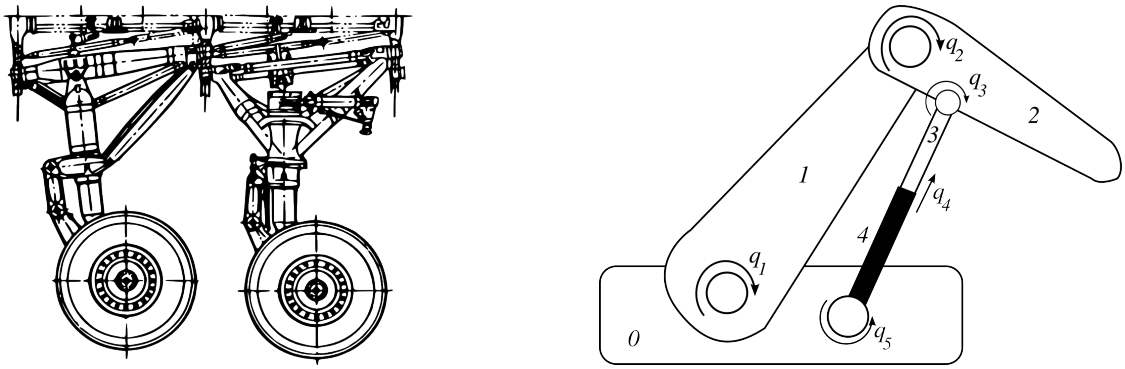

Figure 5.8: Kinematic chains with a graph topology. Left: landing gear of an airplane, with dozens of prismatic and revolute joints interconnected in multiple loops. Right: excavator, with on loop containing five segments (*0* to *4*) and five joints (four revolute joints q_1, q_2, q_3, q_5, and one prismatic joint q_4 in the hydraulic cylinder). The landing gear image is taken from the Open Clip Art Library project, where it is available under a Creative Commons Public Domain license.

TODO: concepts of incidence and adjacency graphs, and their matrix representations; how to "cut" graphs in order to get a tree (for computational purposes).

5.4 Segment-to-segment recursion: position, velocity, acceleration

This Section introduces a set of primitive algorithms that are used in most *numerical solvers* for kinematic chains of generic types, that is, algorithms that make no use of specific geometric properties that can simplify the calculations. (Chapters 7, 8 and 10 give examples of kinematic chains with specific geometries.) The presented algorithms are generic (or "primitive"), since solvers use them as *segment-to-segment recursions* over serial sub-structures of kinematic chains. Such recursive algorithms are popular, since they can achieve *linear-time* complexity: they have a computational load that increases only linearly with the number of joints in the robot. They are typically

used in so-called *inward and outward* recursions (or "*sweeps*") from segment to segment, to propagate position, velocity, acceleration, force, and inertia from the root to the leafs (*outward* recursion), or vice versa (*inward* recursion).

The terminology "inward" and "outward" is, strictly speaking, only unambiguous for classical robot arms: their base 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" base from one foot to the other when walking, and sometimes they have none of their feet on the ground. They can also "climb" such that one of their hands serves as the fixed base, especially in outer space where gravity is absent. Anyway, the topology of a humanoid robot is a *tree*, which allows any of its segments to be taken as the base ("root") of its kinematic chain structure. Without any loss of generality, the current context makes the assumption that the base is fixed in space.

In this text, the recursion step towards segment i means that all physcial properties of interest are expressed with respect to the *reference point* at the origin of the proximal frame $\{p_i\}$ of that segment. Every recursion step consists of two or three sub-steps. For example, an outward recursion from (the proximal frame p_i on) segment i to (the proximal frame p_{i+1} on) segment $i+1$ (Fig. 5.6), requires the following operations:

1. a *change in reference point* from the proximal frame $\{p_i\}$ of segment i to this segment's distal frame $\{d_i\}$. This changes the *vector representation* of the physical entity being transformed (pose, velocity, acceleration, force).

2. the incorporation of the *physical contribution* (position, velocity, acceleration, force, inertia) at joint q_{i+1}, and using the origin of the distal frame $\{d_i\}$ as reference.

3. a *coordinate transformation* of the result from the distal frame $\{d_i\}$ of segment i to the proximal frame $\{p_{i+1}\}$ rigidly attached to the next segment $i+1$.

The last step changes only the *coordinate representation*, not the physical vectors, and it is only necessary if the coordinates are expressed with respect to the proximal frames on the segment; in the case of expressing all coordinates with respect to the fixed world frame, this step is not required.

For efficiency reasons, linear-time algorithms try to keep these three steps as computationally efficient as possible. That is the reason to choose coinciding proximal and distal frames on subsequent segments, up to a motion about the joint axis, which has been chosen to be the Z axis of the local frames, for the single degrees-of-freedom joints considered in this Section. Indeed, for simplicity but without loss of generality, this document assumes that both segments can only move with respect to each other by a *rotation* about $\boldsymbol{Z}_{i+1} = \boldsymbol{Z}_{d_{i+1}} = \boldsymbol{Z}_{p_{i+1}}$. The extension to a pure *translation* is simple, but the extension to joints with more than one single degree-of-freedom is more complex. (TODO: explain these extensions!)

When the rigid body inertias of the segments are also included in a recursion (see Chap 6), it is not uncommon to introduce the *centroidal frame* (Sect. 4.12.4) as another reference frame on each segment; in that frame, the inertia matrix of the local segment has the simplest possible coordinate expression, which allows to reduce the number of calculations.

5.4.1 Outward position recursion

This recursion is just a special case of the composition of frame poses, given by Eq. (4.4): the position and orientation of (the proximal frame on) segment $i+1$ with respect to (the proximal frame on) segment i is a simple function of the measured joint angle. The three-step recursion mentioned in the previous Section reduces to two steps, since steps 2 and 3 are the same in this case where the "physical contribution" *is* the coordinate transformation generated by the motion:

$$ {}^{p_{i+1}}_{p_i}\boldsymbol{T} = {}^{d_i}_{p_i}\boldsymbol{T}\ {}^{p_{i+1}}_{d_i}\boldsymbol{T}(q_i). \tag{5.9} $$

The *branching*—that occurs in a tree-structured chain at segments from which two or more serial chains emerge—does not complicate the motion recursion at all: each outward recursion path remains exactly equivalent to a serial robot, since there is no interaction between two branches. (This will not be the case anymore when dynamics are to be taken into account, Chap. 6). However, the *loops*—that occur in kinematic chains of the graph type—*do* disturb the position recursion, since the influence of one joint in the loop cannot be decoupled from the influence of the other joints in the loop.

5.4.2 Outward velocity recursion

The velocity recursion finds the linear and angular velocity $\dot{\boldsymbol{X}}_{i+1} = (\boldsymbol{\omega}_{i+1}^T, \boldsymbol{v}_{i+1}^T)^T$ of segment $i+1$ (with the origin of the proximal frame $\{p_{i+1}\}$ as velocity reference point, Sect. 4.2.3), given the linear and angular velocity $\dot{\boldsymbol{X}}_i = (\boldsymbol{\omega}_i^T, \boldsymbol{v}_i^T)^T$ of segment i (with the origin of the proximal frame $\{p_i\}$ as velocity reference point), and given the joint angle q_{i+1} and the joint angle speed $\dot{q}_{i+1}$ between both segments.

While a *point* can only have a translational velocity, represented by a three-dimensional vector $\boldsymbol{v}$, and a translational acceleration $\boldsymbol{a} = \dot{\boldsymbol{v}}$, the velocity of a *rigid body* contains both translational and angular velocity components, $\boldsymbol{v}$ and $\boldsymbol{\omega}$; similarly for the body's acceleration $\boldsymbol{a}$: it has a translational component $\dot{\boldsymbol{v}}$ and an angular component $\dot{\boldsymbol{\omega}}$. A correct interpretation of these vector *representations* requires the knowledge of the *(velocity) reference point* on the moving rigid body for which the translational velocity $\boldsymbol{v}$ or translational acceleration $\dot{\boldsymbol{v}}$ are considered (the angular components are not influenced by the choice of reference point). So, the transformation of the *physical* twist component vectors when changing the velocity reference point from the origin of frame $\{i\}$ to the origin of frame $\{i+1\}$ are:

$$\boldsymbol{\omega}_{i+1} = \boldsymbol{\omega}_i, \qquad \boldsymbol{v}_{i+1} = \boldsymbol{v}_i + \boldsymbol{r}^{i+1,i} \times \boldsymbol{\omega}_i, \tag{5.10}$$

with $\boldsymbol{r}^{i+1,1}$ the position vector from the origin of frame $\{i+1\}$ to the origin of frame $\{i\}$. When expressing $\dot{\boldsymbol{X}}_i$ and $\dot{\boldsymbol{X}}_{i+1}$ in their local coordinate frames (that is, the proximal segment frames $\{i\}$, respectively $\{i+1\}$), the *coordinate* expression of the velocity recursion is:

$$\dot{\boldsymbol{X}}_{i+1} = \begin{pmatrix} \boldsymbol{\omega}_{i+1} \\ \boldsymbol{v}_{i+1} \end{pmatrix} = {}_{i+1}^{i}\boldsymbol{S} \begin{pmatrix} \boldsymbol{\omega}_i \\ \boldsymbol{v}_i \end{pmatrix} = {}_{i+1}^{i}\boldsymbol{S}\, \dot{\boldsymbol{X}}_i. \tag{5.11}$$

with ${}_{i+1}^{i}\boldsymbol{S}$ the screw transformation matrix from the proximal frame on segment i to the proximal frame on the distal segment $i+1$. It was defined in Eq. (2.27), and is repeated here for convenience:

$$\begin{pmatrix} {}_{i+1}^{i}\boldsymbol{R} & \boldsymbol{0}_{3\times 3} \\ \left[{}_{i+1}\boldsymbol{r}^{i+1,i}\right] {}_{i+1}^{i}\boldsymbol{R} & {}_{i+1}^{i}\boldsymbol{R} \end{pmatrix}, \tag{5.12}$$

where "$[\boldsymbol{r}]$" stands for "taking the vector product with $\boldsymbol{r}$".

The *order* of the linear and angular three-dimensional vectors $\boldsymbol{v}$ and $\boldsymbol{\omega}$ in Eq. (5.11) is *arbitrary*, Sect. 4.2.2. Making the alternative choice, $\dot{\boldsymbol{X}}_i = (\boldsymbol{v}_i^T\, \boldsymbol{\omega}_i^T)^T$ implies that the coordinate transformation formulae have to be changed accordingly.

$$\begin{pmatrix} \boldsymbol{v}_{i+1} \\ \boldsymbol{\omega}_{i+1} \end{pmatrix} = \boldsymbol{\Delta}\ {}_{i+1}^{i}\boldsymbol{S}\ \boldsymbol{\Delta} \begin{pmatrix} \boldsymbol{v}_i \\ \boldsymbol{\omega}_i \end{pmatrix}, \tag{5.13}$$

with Δ as in Eq. (2.19), and repeated here for convenience:

$$\boldsymbol{\Delta} = \begin{pmatrix} \boldsymbol{0}_3 & \boldsymbol{I}_3 \\ \boldsymbol{I}_3 & \boldsymbol{0}_3 \end{pmatrix}. \tag{5.14}$$

This alternative choice in ordening $\boldsymbol{v}$ and $\boldsymbol{\omega}$ will *not* be used furtheron in this text, unless noted explicitly.

The segment-to-segment velocity recursion which is the subject of this Section is *more* than just the *transformation* of velocity reference point: the joint velocty $\dot{q}_{i+1}$ *adds* a contribution (around the local $\boldsymbol{Z}_{i+1}$-axis) to the angular velocity of segment $i+1$, so the following recursion equations are an obvious extension of the pure velocity transformation Eq. (5.10):

$$\boldsymbol{\omega}_{i+1} = \boldsymbol{\omega}_i + \dot{q}_{i+1}\boldsymbol{Z}_{i+1}, \quad \boldsymbol{v}_{i+1} = \boldsymbol{v}_i + \boldsymbol{r}^{i+1,i} \times \boldsymbol{\omega}_i. \tag{5.15}$$

Or, in case joint $i+1$ is a translational joint along the $\boldsymbol{Z}_{i+1}$-axis:

$$\boldsymbol{\omega}_{i+1} = \boldsymbol{\omega}_i, \quad \boldsymbol{v}_{i+1} = \boldsymbol{v}_i + \dot{q}_{i+1}\boldsymbol{Z}_{i+1} + \boldsymbol{r}^{i+1,i} \times \boldsymbol{\omega}_i. \tag{5.16}$$

In coordinate form, the recursion $\dot{\boldsymbol{X}}_i \to \dot{\boldsymbol{X}}_{i+1}$ becomes:

$$\dot{\boldsymbol{X}}_{i+1} = {}_{i+1}^{i}\boldsymbol{S}\dot{\boldsymbol{X}}_i + \dot{q}_{i+1}\boldsymbol{Z}_{i+1}, \tag{5.17}$$

The joint axis six-vector $\boldsymbol{Z}_{i+1}$ has a simple coordinate expression when it lies along the Z axis of the proximal segment frame $\{p_{i+1}\}$:

$$\boldsymbol{Z}_{i+1} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}^T \tag{5.18}$$

for a rotational joint, and

$$\boldsymbol{Z}_{i+1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}^T \tag{5.19}$$

for a translational joint. Recall that these expressions depend on the *arbitrary choice* made in this document (Sect. 4.2.2) to put the angular velocity coordinates *on top*, and the linear velocity coordinates at the *bottom* of the six-vector of twist coordinates.

5.4.3 Outward acceleration recursion

This recursion calculates the linear and angular acceleration $\ddot{\boldsymbol{X}}_{i+1} = (\dot{\boldsymbol{\omega}}_{i+1}^T, \dot{\boldsymbol{v}}_{i+1}^T)^T$ (with the origin of the proximal frame $\{i+1\}$ as velocity reference point, Sect. 4.2.3), given the linear and angular acceleration $\ddot{\boldsymbol{X}}_i = (\dot{\boldsymbol{\omega}}_i, \dot{\boldsymbol{v}}_i)$ (with the origin of the proximal frame $\{i\}$ as velocity reference point), and given the joint angle q_{i+1}, the joint angle speed $\dot{q}_{i+1}$, and the joint acceleration $\ddot{q}_{i+1}$. The recursion equations (for the rotational joint case) are found straightforwardly by taking the time derivative of the *physical* velocity recursion in Eq. (5.15):

$$\dot{\boldsymbol{\omega}}_{i+1} = \dot{\boldsymbol{\omega}}_i + \ddot{q}_{i+1}\boldsymbol{Z}_{i+1} + \dot{q}_{i+1}\boldsymbol{\omega}_i \times \boldsymbol{Z}_{i+1}, \tag{5.20}$$

$$\dot{\boldsymbol{v}}_{i+1} = \dot{\boldsymbol{v}}_i + \boldsymbol{r}^{i+1,i} \times \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times (\boldsymbol{r}^{i+1,i} \times \boldsymbol{\omega}_i). \tag{5.21}$$

This uses the property that $\boldsymbol{\omega} \times \boldsymbol{x}$ is the time derivative of a vector $\boldsymbol{x}$ that is fixed to a body that rotates with an angular velocity $\boldsymbol{\omega}$. In *coordinate* matrix form, the recursion $\ddot{\boldsymbol{X}}_i \to \ddot{\boldsymbol{X}}_{i+1}$ becomes:

$$\begin{aligned} \ddot{\boldsymbol{X}}_{i+1} &= {}^{i}_{i+1}\boldsymbol{S}\left(\ddot{\boldsymbol{X}}_i + \ddot{\boldsymbol{X}}_{b,i}\right) + \ddot{q}_{i+1}\boldsymbol{Z}_{i+1}, \\ \text{with} \quad \ddot{\boldsymbol{X}}_{b,i} &= \begin{pmatrix} \boldsymbol{\omega}_i \times \dot{q}_{i+1}\,\boldsymbol{Z}_{i+1} \\ \boldsymbol{\omega}_i \times ({}_i\boldsymbol{r}^{i+1,i} \times \boldsymbol{\omega}_i) \end{pmatrix}. \end{aligned} \tag{5.22}$$

(As for the velocity case (Sect. 5.4.2), all coordinates are expressed with respect to the *proximal segment frames* $\{p_i\}$ and $\{p_{i+1}\}$.) The acceleration recursion is similar to the velocity recursion of Eq. (5.17), except for the *"bias acceleration"* $\ddot{\boldsymbol{X}}_{b,i}$ of the moving frame $\{p_{i+1}\}$ due to the non-vanishing angular velocity $\boldsymbol{\omega}_i$ of segment i. This is the *gyroscopic acceleration*, which is not physical (in the sense that it cannot be used to generate power, for example) but just an artifact of expressing coordinates in a non-inertial reference frame.

5.5 Jacobian matrix

This Section discusses the concept of the Jacobian matrix of a kinematic chain. The concept is important because it provides a *linearization* of the, in general, nonlinear motion properties of a kinematic chain—represented by Eqs. (5.1)–(5.2)—and hence, the Jacobian matrix is a key component in the analysis and computations of all *instantaneous* motion properties of kinematic chains: position closure (Sect. 5.5.4), velocity transformations between joints and end-effectors (this Section), numerical detection and analysis of kinematic singularities (Sect. 5.8.2), and redundancy resolution (Sect. 5.9).

In a general kinematic chain, moving one joint and keeping all others motionless results in a motion of the end-effector(s) of the chain that is a *nonlinear* function of the joint motion. However, *the limit to zero* of this motion, or, alternatively, the relationship between the *joint velocity* and the *end-effector velocity* are *linear*. And linear relationships are always computationally more interesting to work with than nonlinear relationships. They are particularly interesting for the realtime end-effector motion control of robots, where one needs to calculate the joint velocity that *instantaneously* moves the end-effector in the required direction and with the required speed.

Serial chains are the simplest to define and calculate the Jacobian, and that serial definition will be reused for tree and graph structures, since both can be considered as a set of serial chains linked together at some segments in the chain. The major difference between serial structures and tree structures is that, in a serial structure, every joint influences the motion of the chain's end-effector (at least, in the purely kinematics context, i.e., without considering dynamic effects), while in a tree structure, some joints do not influence the motion of some of the multiple end-effectors. The major difference between serial and tree structures on the one hand, and graph structures on the other hand, is that, in the latter, the motion of one joint *can be constrained* by the motion of the other joints, via the physical "loops" in the structure.

5.5.1 Jacobian matrix for serial chains

In a general serial chain, the input-output relation from joint positions $\boldsymbol{q}$ to end-effector pose(s) $\boldsymbol{T}$ is nonlinear (and uniquely defined), but the relationship between the joint velocities $\dot{\boldsymbol{q}}$ and the end-effector(s) twist $\mathbf{t}^{ee}$ is *linear*: if one drives a joint twice as fast, the end-effector will move twice as fast too. Hence, the linear relationship can be *represented* by a *matrix*:

$$\underset{6\times 1}{{}_{bs}\mathbf{t}^{ee}} = \underset{6\times n}{{}_{bs}\boldsymbol{J}(\boldsymbol{q})} \quad \underset{n\times 1}{\dot{\boldsymbol{q}}}. \tag{5.23}$$

The matrix ${}_{bs}\boldsymbol{J}(\boldsymbol{q})$ is called the *Jacobian matrix*, or *Jacobian* for short, with respect to the reference frame $\{bs\}$. This matrix is a nonlinear function of the joint positions $\boldsymbol{q}$. Hence the notation $\boldsymbol{J}(\boldsymbol{q})$ used above, but one most often omits this explicit mention of $\boldsymbol{J}$'s dependence on the joint positions $\boldsymbol{q}$.

The Jacobian has a clear *physical interpretation*: column i of $\boldsymbol{J}$ is the end-effector velocity (or "twist" $\mathbf{t}_i$) that results from giving a unit velocity to joint i, and a zero velocity to all other joints. Hence, the whole Jacobian matrix gives a *(coordinate) basis* for the vector space of all possible end-effector twists; and an individual column of the Jacobian is sometimes called a *partial* twist, [165]. The twist interpretation of the Jacobian implies that the joint rates $\dot{\boldsymbol{q}}$ in Eq. (5.23) are *dimensionless* coordinates: they represent the *magnitude* of the twist with respect to the (arbitrarily defined) "unit twist." While the mapping from joint velocities to end-effector motion is a uniquely defined *physical* concept, different Jacobian *matrices* (i.e., coordinate *representations*) exist, because the six-dimensional end-effector velocities $\mathbf{t}_i$ have various coordinate representations (Sect. 4.2.2), depending on:

- the reference frame $\{bs\}$ with respect to which the end-effector twist $\mathbf{t}^{ee}$ is expressed.
- a number of arbitrary choices for the coordinates of the twist on the left-hand side of Eq. (5.23): angular velocity on top, or linear velocity on top (Eqs. (4.12) and (4.13)), and physical units of angular and linear velocity components.
- the option to take the velocity reference point of the twist in the origin of the base frame ("screw twist") or in the origin of the end-effector frame ("pose twist").

The choice between pose or screw twists has an important practical consequence: for pose twists, each column in the Jacobian depends on *all* joint positions, while for screw twists, only the proximal joints have an influence. The reason is that pose twists require the knowledge of where (a reference point on) the *end-effector* is located, while the screw twist only requires the knowledge of the relative position and orientation between the joint and the origin of the *base* reference frame. And since the position of the end-effector is influenced by the positions of *all* joints, the columns in "pose twist" Jacobians depend on all joint positions.

Units of twists are *not* homogeneous: three of them have the units of angular velocity (radians per seconds, or an equivalent choice, such as degrees per seconds), the three others have the units of linear velocity (meter per second, or an equivalent choice, such as inches per second). So, there is no such thing as the "Euclidean norm" of a twist, Sect. 4.9, or the "Euclidean distance" between two twists. Hence, all "measures" based on the Jacobian matrix are to be investigated with some caution, because they could lack physical interpretation. One such measure, used in many papers in the literature, is the *manipulability index* $\boldsymbol{J}^T\boldsymbol{J}$, [70, 176, 195, 243, 288]; it should measure how "easy" or "difficult" it is for a robot to move in a given Cartesian direction. However, filling in the physical units of the Jacobian matrix reveals that this concept can not have any physical meaning since the units do not match... In addition, the "measured" value depends on the choice of units in the angular and linear components of the twists, and on the reference frame in which they are expressed. The only "solution" to this *non-invariance* problem is to introduce a "weighting matrix", with matching physical units, such as in $\boldsymbol{J}^T\boldsymbol{M}\boldsymbol{J}$. The interpretation of $\boldsymbol{M}$ is that it is an *operator* from twist space to wrench space (Sect. 4.12.10), and back, as is clear from its position in the equation: $\boldsymbol{J}$ maps a joint velocity into a Cartesian velocity, and $\boldsymbol{J}^T$ maps a Cartesian force into joint forces. (Check the physical units and this becomes clear!). Hence, the weighting matrix must map a Cartesian velocity (or displacement, or acceleration, since these only differ by one order of time derivative more or less) into a Cartesian force, hence, it has the physical meaning of damping, stiffness or inertia. For example, the *dynamic manipulability* is a measure for how much inertia the robot feels when moving in a certain Cartesian direction.

5.5.2 Analytical Jacobian

The concept of the robot Jacobian was introduced by Withney, [276], but [201] presents already an earlier similar coordinate description, without using the name "Jacobian". The terminology is inspired by the "Jacobian matrix"

as defined in classical mathematical analysis, (i.e., the matrix of partial derivatives of a function, [33, 52, 156, 222, 247]) named after the Prussian mathematician Karl Gustav Jacob Jacobi (1804–1851). However, the Jacobian matrix used in robotics has, most often, not exactly the same meaning as this analytical Jacobian: the angular velocity three-vector $\boldsymbol{\omega}$ is not the time derivative of any three-vector orientation representation. Nevertheless, the time derivative of the explicit (forward) position kinematic function $\boldsymbol{X} = f(\boldsymbol{q})$, Eq. (5.1), is well-defined:

$$\dot{\boldsymbol{X}} = \frac{d\boldsymbol{X}}{dt} = \sum_{i=1}^{n} \frac{\partial f(\boldsymbol{q})}{\partial q_i} \frac{\partial q_i}{\partial t} = \widehat{\boldsymbol{J}}\dot{\boldsymbol{q}}. \tag{5.24}$$

Most often, a *minimal representation* for the end-effector pose $\boldsymbol{X}$ is chosen, in the form of a finite displacement twist. The angular coordinates of the finite displacement twist $\mathbf{t}_d$ are a set of three Euler angles. The time derivatives of these Euler angles are related to the angular velocity three-vector by means of *integrating factors*. These integrating factors constitute the difference between the Jacobian in Eq. (5.23) and the matrix of partial derivatives $\widehat{\boldsymbol{J}} = \partial f(\boldsymbol{q})/\partial q_i$ in Eq. (5.24). The latter one is sometimes called the *analytical Jacobian*, [95, 233, 131], when it is necessary to distinguish it from the *twist Jacobian* $\boldsymbol{J}$ in Eq. (5.23).

Most literature uses both notations $\dot{\boldsymbol{X}}$ and $\mathbf{t}$ interchangebly, and this text will do also when there is no need to distinguish explicitly between the twist version and the analytical version of the joint velocities to end-effector velocity mappings.

5.5.3 Jacobian matrix for tree structures

The major differences between a serial chain and a tree chain are that (i) the latter has multiple end-effector segments, and (ii) some joints do not influence the motion of all end-effectors. Figure 5.1 shows an example of such a chain, with one base (with reference frame {0}), two end-effectors (with reference frames {1} and {2}), and five revolute joints. Formally, Eq. (5.23) keeps the same structure for a tree chain as for a serial chain, but, for a tree with N end-effectors, the "twist" on the left-hand side of Eq. (5.23) is a $6N \times 1$ vector, and the Jacobian matrix is a $6N \times N$ matrix:

$$\begin{pmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \vdots \\ \mathbf{t}_N \end{pmatrix} = \begin{pmatrix} \boldsymbol{J}_1^1 & \boldsymbol{J}_1^2 & \dots & \boldsymbol{J}_1^N \\ \boldsymbol{J}_2^1 & \boldsymbol{J}_2^2 & \dots & \boldsymbol{J}_2^N \\ \dots & \dots & \dots & \dots \\ \boldsymbol{J}_N^1 & \boldsymbol{J}_N^2 & \dots & \boldsymbol{J}_N^N \end{pmatrix} \begin{pmatrix} \dot{\boldsymbol{q}}_1 \\ \dot{\boldsymbol{q}}_2 \\ \vdots \\ \dot{\boldsymbol{q}}_N \end{pmatrix} \tag{5.25}$$

Formally, this relationship can still be expressed in the form of Eq. 5.23:

$$\mathbf{t} = \boldsymbol{J}\dot{\boldsymbol{q}}, \tag{5.26}$$

but, both, the end-effectors twist $\mathbf{t}$ and the Jacobian matrix $\boldsymbol{J}$, now have somewhat broader interpretations. In general, the $6N \times N$ Jacobian matrix has many zero submatrices $\boldsymbol{J}_{(6-1)j\dots(6-1)(j+1),i\dots i+k}$ on the places where joints i to $i+k$ do not move the jth end effector. For the example in Fig. 5.1, this would give the following:

$$\begin{pmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{J}_1 & \boldsymbol{J}_2 & \boldsymbol{J}_3 & \mathbf{0} & \mathbf{0} \\ \boldsymbol{J}_1 & \mathbf{0} & \mathbf{0} & \boldsymbol{J}_4 & \boldsymbol{J}_5 \end{pmatrix} \dot{\boldsymbol{q}}, \tag{5.27}$$

with $\mathbf{0}$ a 6×1 vector of zeroes. In full coordinate form, this yields:

$$\begin{pmatrix} \mathsf{t}_{11} \\ \mathsf{t}_{12} \\ \mathsf{t}_{13} \\ \mathsf{t}_{14} \\ \mathsf{t}_{15} \\ \mathsf{t}_{16} \\ \mathsf{t}_{21} \\ \mathsf{t}_{22} \\ \mathsf{t}_{23} \\ \mathsf{t}_{24} \\ \mathsf{t}_{25} \\ \mathsf{t}_{26} \end{pmatrix} = \begin{pmatrix} J_{11} & J_{21} & J_{31} & 0 & 0 \\ J_{12} & J_{22} & J_{32} & 0 & 0 \\ J_{13} & J_{23} & J_{33} & 0 & 0 \\ J_{14} & J_{24} & J_{34} & 0 & 0 \\ J_{15} & J_{25} & J_{35} & 0 & 0 \\ J_{16} & J_{26} & J_{36} & 0 & 0 \\ J_{11} & 0 & 0 & J_{41} & J_{51} \\ J_{12} & 0 & 0 & J_{42} & J_{52} \\ J_{13} & 0 & 0 & J_{43} & J_{53} \\ J_{14} & 0 & 0 & J_{44} & J_{54} \\ J_{15} & 0 & 0 & J_{45} & J_{55} \\ J_{16} & 0 & 0 & J_{46} & J_{56} \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \end{pmatrix}.$$

$\mathtt{t}_{1i}$ are the six coordinates of the velocity of end-effector 1, only influenced by the first three joints, and $\mathtt{t}_{2i}$ are the six coordinates of the velocity of end-effector 2, only influenced by the joints 1, 4 and 5.

The first column contains twice the same six-dimensional vector, describing the influence of the first joint velocity on *both* the end-effector velocities. This is only the case for *screw twists*, Sect. 4.2.2, since the velocity reference point for both end-effectors is the same, namely the (instantaneous) origin of the base frame.

5.5.4 Jacobian matrix for chain structures

Chain structures have one property that serial and tree chains do not have: they have at least one loop. While the motion of a joint in a serial or tree chain is not constrained by the motion of the other joints, this is not the case anymore in a loop. Each loop has to satisfy *loop closure* constraints for position as well as velocity (and all higher order derivatives):

- *Position closure*: at any time, the joint positions $\boldsymbol{q}$ and end-effector poses $\boldsymbol{X}$ must satisfy the constraint in Eq. (5.1), that is, there should be no "gaps" between joints and segments in the *mathematical model* of the kinematic chain. The real physical chain will, of course, never have such gaps, unless it has broken.

- *Velocity closure*: the velocity around the complete loop must vanish. The example in Fig. 5.7 shows a graph chain, with a loop over joints 2, 3, 4 and 6. Consider the serial sub-chain that has its (arbitrarily chosen) "base" at joint 2, runs over joint 3, then joint 6, joint 4, and finally with its "end-effector" at joint 2 again, coinciding with the "base". Then, the following Jacobian equation is the mathematical representation of the loop closure constraint that holds for all possible joint values:

$$\begin{pmatrix} \boldsymbol{J}_2 & \boldsymbol{J}_3 & \boldsymbol{J}_6 & \boldsymbol{J}_4 \end{pmatrix} \begin{pmatrix} \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_6 \\ \dot{q}_4 \end{pmatrix} = \boldsymbol{0}, \tag{5.28}$$

  where $\boldsymbol{J}_i$ is the local loop's 6×1 "end-effector" velocity generated by a unit velocity at joint i, if no loop would exist to constrain the influence of that joint i on the end-effector. This closure equation clearly puts constraints on the allowable *velocities* at the joints in the loop, *and* this constraint is configuration dependent since the Jacobian columns depend on the *positions* of all joints in the loop.

In the closure constraint relationships, the joint positions and velocities are the unknowns that have to be calculated, while the geometric dimensions (distances, orientations) of segments and joints are fixed parameters. The velocity closure can only be solved as soon as the joint positions are are known, since the columns of the Jacobian matrix depend on those joint positions. Some joint positions (typically those of the actuated joints) are known because they are *measured*, and hence do not have to be calculated from the loop closure. (In some rare cases, there are also measurements available of frames on the kinematic chain; for example, provided by an external optical measurement system.) Some (most often passive) joints have no position *sensors*, hence those joint positions much be *calculated* by solving the (nonlinear!) Eq. (5.1) with the fixed parameters and the measured joint positions as inputs. For *serial* or *tree* chains, the position "closure" is trivial (at least in the typical case that all joints have position sensors) since it is not a real closure constraint at all: no joint constraints the motion of any other joint, and one set of measured joint positions always corresponds to only one single pose for all end-effectors, (Sect. 5.4.1). However, the solution is, in general, not unique, in chains with a number of joints *without* position sensors, such as in chains with a graph structure. The most common calculation approach for the position closure is:

1. To "cut" (the model of) the graph chain into a tree. This is illustrated in Fig. 5.7: the loop over the segments connected by joints 2, 3, 4 and 6 is cut between joints 3 and 6, giving rise to two extra *virtual* "end-effectors" with frames $\{c_1\}$ and $\{c_2\}$.

2. To calculate the pose of all cut "end-effector" frames in the tree based on the measured joint positions and estimates for the passive joint positions.

3. To calculate a joint position increment that makes the cut frames move closer to each other, reducing the gap between them. This calculation makes use of a the Jacobian matrix of the cut tree chains: one chooses an "end-effector" twist for the cut loop that reduces the gap, and calculates the corresponding joint increments from an equation of the form as Eq. (5.27). (In fact, this gap closure problem is the *inverse*

velocity kinematics problem of the (virtual) cut chain. Later Sections explain the details of such IVK solvers.)

Because the columns of the Jacobian depend on the solution of the position closure, *and* because the Jacobian is only a first-order, linearized version of the motion properties of the chain, and hence the gap will seldom be closed in one single step, closure constraints are almost always solved *iteratively*. The algorithm also involves quite a lot of *arbitrary* choices: Where to cut a loop? What end-effector twist to choose to make the cut gap smaller? When to stop the iteration, that is, when is the (virtual) gap "small enough"?

So, with the position and velocity closures solved, the concept of the Jacobian matrix for the *whole* graph kinematic chain can be defined. For example, in the case of Fig. 5.7, the total mapping from the six joint velocities $\dot{q}_i$ to the two end-effector velocities $\mathbf{t}_1$ and $\mathbf{t}_2$ is still given by a linear "Jacobian mapping":

$$\underbrace{\begin{pmatrix}\mathbf{t}_1\\ \mathbf{t}_2\\ \mathbf{0}\end{pmatrix}}_{\mathbf{t}} = \underbrace{\begin{pmatrix}\boldsymbol{J}_1 & \boldsymbol{J}_2 & \boldsymbol{J}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \boldsymbol{J}_1 & \mathbf{0} & \mathbf{0} & \boldsymbol{J}_4 & \boldsymbol{J}_5 & \mathbf{0}\\ \mathbf{0} & \boldsymbol{J}_2 & \boldsymbol{J}_3 & \boldsymbol{J}_4 & \mathbf{0} & \boldsymbol{J}_6\end{pmatrix}}_{\boldsymbol{J}} \underbrace{\begin{pmatrix}\dot{q}_1\\ \dot{q}_2\\ \vdots\\ \dot{q}_6\end{pmatrix}}_{\dot{\boldsymbol{q}}}. \tag{5.29}$$

Of course, the graph chain shown in Fig. 5.7 is just an artificial example, with less useful structure than chain graph designs that people have designed for real-world tasks. Figure 5.9 shows a more realistic example, of a so-called *parallel robot*, Chap. 8. That structure has six "legs", providing five (parallel) loops between end-effector and base; it has six *actuated* joints (driving the prismatic joints in the legs) and all the rest are *passive* revolute joints. This gives this structure six degrees-of-freedom in total.

The previous paragraph already briefly mentioned the concept of passive and active joints. This concept is most often only relevant for chain structures, since serial and tree structures would collapse if one or more of the joints would not be actively driven by a motor. On the other hand, it is exceptional that graph chains have as many motors as joints, since the loop closure relationships allow "to drive" the passive joints via the active joints. Most often, the Jacobian relationships involve the *active* joints only, but this practice hides some potential problems, e.g., to analyse *kinematic singularities*, Sect. 5.8.

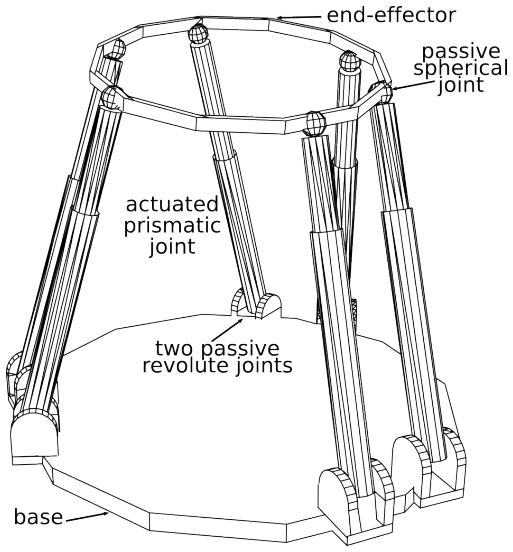

Figure 5.9: A *parallel robot* with five kinematic loops, and with a particular mechanical design: six actuated prismatic "legs" connect the base and end-effector platforms, via spherical and/or "gimbal" joints at both ends. This design is often named after *Stewart* and *Gough*.

5.6 Accuracy, repeatability, and calibration

Finding the pose and velocity of the end-effector given the actual joint positions and velocities relies on a *mathematical idealization*. Such mathematical models are never 100% accurate, due to: tolerances introduced at

manufacturing or due to wear; elasticity in links, joints and transmissions; friction in joints, transmissions and external contacts; backlash in joints or transmissions; finite accuracy with which joint positions, velocities and torques can be measured, or with which all geometric and dynamic parameters can be identified. This means that the real end-effector motion differs from the modelled one. The smaller this difference, the better the *absolute (positioning) accuracy* of the robot, i.e., the mean difference between the actual motion and the motion 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.

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.

5.6.1 Off-line programming — 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, [19, 105, 168, 259, 233]:

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 are *arbitrarily chosen* and for a *model* to explain 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, Sect. 5.3.1.

Step 2 (Data collection). The robot is moved to a large set of N different poses where its end-effector homogeneous transform ${}^{ee}_{bs}\boldsymbol{T}_m(\boldsymbol{q}_i), i = 1, \ldots, N$ 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), [19].

Step 3 (Parameter fitting). The real poses are expressed as a Taylor series in the error parameters $P_j, j = 1, \ldots, n$:

$$ {}^{ee}_{bs}\boldsymbol{T}(\boldsymbol{q}_i, P) = {}^{ee}_{bs}\boldsymbol{T}(\boldsymbol{q}_i, 0) + \sum_{j=1}^{n} \left\{ \partial \left({}^{ee}_{bs}\boldsymbol{T}(\boldsymbol{q}_i, P) \right) / \partial P_j \right\} \; P_j + \mathcal{O}(P^2). \tag{5.30} $$

The first term in this series is the pose derived from the model; the second term represents the *first-order* (or *linearized*) effect of the error parameters on the kinematic transformation. The error parameters could be considered as extra "joints" in the kinematic chain, so the above-mentioned second term is nothing else but the Jacobian equation, Eq. (5.23), of the kinematic chain extended with the virtual error parameter joints, [168, p. 110].

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

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 often in a rather limited subset of the robot's workspace only. Due to the presence of the error parameters, a calibrated robot always has a *general* kinematic structure, even if its nominal model is of a special geometric type (such as the very common serial 321 type, Sect. 7.2). Hence: (i) calibrated robots definitely need the *numerical* kinematic solvers described in this and some of the following Chapters, and (ii) they have a good chance of hitting the *representation singularities* of minimal frame conventions such as Denavit-Hartenberg (Sect. 5.3.1) and Hayati-Roberts (Sect. 5.3.2).

5.7 Generic operations on kinematic chains

The following Sections define these transformation problems in a formal, generic way; subsequent Chapters then introduce concrete algorithms to solve these transformations, in ways that are fast enough to be used on-board of robots, in realtime, and that take into account the particular properties of specific kinematic chain structures. Indeed, the kinematic model of the chain plays a prominent role in all transformation operations: it contains the information about which connections exist between segments and joints, and at which position and orientation exactly the joints are connected to the segments.

5.7.1 Forward position kinematics

The forward position kinematics (FPK) of a kinematic chain with n joints solves the following problem:

> *Given the joint positions* $\boldsymbol{q} = (q_1 \ \ldots \ q_n)^T$*, and the model of the kinematic chain, what are the corresponding end-effector poses?*

5.7.2 Forward velocity kinematics

The forward velocity kinematics (FVK) solves the following problem:

> *Given the vectors of (i) joint positions* $\boldsymbol{q} = (q_1 \ \ldots \ q_n)^T$ *and (ii) joint velocities* $\dot{\boldsymbol{q}} = (\dot{q}_1 \ \ldots \ \dot{q}_n)^T$*, and the model of the kinematic chain, what is the resulting end-effector twist(s)* $\mathbf{t}^{ee}$*?*

The solution is always unique in case all joints are actuated: one given set of joint positions and joint velocities always corresponds to only one single end-effector twist.

5.7.3 Inverse position kinematics

The inverse position kinematics ("IPK") solves the following problem:

> *Given the actual end-effector pose(s)* ${}^{ee}_{bs}\boldsymbol{T}$*, and the model of the kinematic chain, what are the corresponding joint positions* $\boldsymbol{q} = (q_1 \ \ldots \ q_n)^T$*?*

5.7.4 Inverse velocity kinematics

Assuming that the inverse *position* kinematics problem has been solved for the current end-effector pose(s) ${}^{ee}_{bs}\boldsymbol{T}$, the inverse *velocity* kinematics ("IVK") then solves the following problem:

> *Given the end-effector twist(s)* $\mathbf{t}^{ee}$*, and the model of the kinematic chain, what is corresponding vector of joint velocities* $\dot{\boldsymbol{q}} = (\dot{q}_1 \ \ldots \ \dot{q}_n)^T$*?*

5.7.5 Inverse force kinematics

Assuming that the inverse *position* kinematics problem has been solved for the current end-effector pose(s) ${}^{ee}_{bs}\boldsymbol{T}$, the inverse *force* kinematics ("IFK") then solves the following problem:

> *Given the 6D force ("wrench")) $\mathbf{F}^{ee}$ that acts on the end-effector(s), and the model of the kinematic chain, what is the vector of joint forces/torques $\boldsymbol{\tau} = (\tau_1 \ \ldots \ \tau_n)^T$ that keeps $\mathbf{F}^{ee}$ in static equilibrium?*

5.7.6 Forward force kinematics

The forward force kinematics ("FFK") solves the following problem:

> *Given the vectors of joint force/torques $\boldsymbol{\tau} = (\tau_1 \ \ldots \ \tau_n)^T$, and the model of the kinematic chain, what is the resulting static force $\mathbf{F}^{ee}$ that the end-effector(s) exert(s) on the environment?*

This problem is only well-defined if the robot has *six* joints, *and* if the end-effector is rigidly fixed to a rigid environment! If $n < 6$ the robot cannot generate a full six-dimensional space of end-effector forces; if $n > 6$ all forces can be generated in infinitely many ways. In addition, most often, the FFK is a dubious "property" of the robot: it depends equally much on the dynamic properties of the *environment* with which the robot is in contact.

5.7.7 Hybrid kinematics

The hybrid kinematics ("HK") solves the following problem:

> *Giving the motion inputs to some of the joints, force inputs to the other joints, and the model of the kinematic chain, what is the resulting motion of the total chain?*

5.8 Singularities

The purpose of the mechanics of a kinematic chain is to transform force and motion from the motors to the end-effector(s) of the robot. For most chains, these transformations "break down" in some configurations (i.e., sets of joint values) of the chains, in that the mappings between the joint and Cartesian spaces, in either forward or inverse direction, have non-smooth behaviour for some specific sets of joint values: a small change in one of the spaces (e.g., a small displacement of an end-effector) does not correspond to a smooth, small change in the other space. This phenomenon is called a *kinematic singularity*. Another physical manifestation of a singularity is that the number of degrees of motion freedom changes when the chain enters a singular configuration. This can happen in the following ways:

- *Active* degrees of freedom are *lost*: at the singularity, the chain's motors fail *to generate a motion* of the end-effectors that is available outside of the singularity.
- *Passive* degrees of freedom are *gained*: the chain's motors *cannot prevent the chain to move* in directions that are actively controllable outside of the singularity. This type of singularity does not occur in serial and tree chains with only actuated joints.

Mathematically speaking, this change in number of motion degrees of freedom corresponds to a change in the *rank* of the Jacobian matrix. For serial and tree structures, a change in the rank of the Jacobian is related to the singularities of the chains, but for graph chains the situation is a bit more complicated: each of the loops can have its own singular configuration that is not visible in the Jacobian matrix, since that matrix typically represent only the *active joint*-to-end-effector mapping. But also the passive joints have to be taken into account in the singularity analysis, since they can give rise to a change in the overall number of degrees of freedom of the chain, [86]. This can be seen by linearizing the implicit relationship Eq. (5.1) between the motion $\boldsymbol{X}$ of the end-effector(s) and the motion of all (active *and* passive) joints $\boldsymbol{q}$:

$$\boldsymbol{f}(\boldsymbol{X}, \boldsymbol{q}) = 0 \quad \Rightarrow \quad \frac{\partial f}{\partial \boldsymbol{X}}\dot{\boldsymbol{X}} + \frac{\partial f}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} = 0. \tag{5.31}$$

Comparison with the definition of the Jacobian matrix, Eq. (5.23), shows that $\boldsymbol{J} = -(\partial f/\partial \boldsymbol{q})/(\partial f/\partial \boldsymbol{X})$. Each of the $\partial f/\partial \cdot$ parts can loose rank in itself, indicating a singularity in either the "forward" or the "inverse"

kinematic transformations. (Both singularities could occur at the same configuration.) For example, each leg in the *Stewart-Gough* platform of Fig. 5.9 is a six degrees of freedom serial chain (with only one actuated joint) and can have its own singularity; for example, the spherical joint can have the "spherical wrist singularity" described in Sect. 3.13. (Some mechanisms have been designed that have the kinematics of a spherical wrist, but are, in practice, *singularity-free*, [244, 277]: the singularities are designed to lie in a configuration that the chain cannot reach because of collisions between some of its links.)

A singular configuration can be one single combination of joint angles, isolated in the space of all possible joint values. But more often, singular configurations for a connected subset in that space, forming itself a subspace of a lower dimension.

5.8.1 Practical consequences of singularities

In and around its singular configurations, a kinematic chain has a different mechanical behaviour than in the rest of its workspace, since it looses an active degree of motion freedom, and/or gains a passive degree of motion freedom. The consequences on various aspects of a robot system are as follows:

- *Trajectory planning*: when planning motions for a robot, one should always analyse how close the planned motion comes to singular configurations, because the robot will experience one—and probably more than one!—of the problems further down this list.

- *Actuator saturation*: in a singularity, the inverse of the Jacobian does not exist , so the transformation from Cartesian setpoints for the end-effector to the joint space joint setpoints "goes to infinity" and even a small desired Cartesian velocity of an end-effector is transformed into an extremely large requested joint velocity. This is of course not possible in practice, since the power amplifier of the actuators will reach its saturation limit. Hence, the real joint velocity will be lower than the calculated one. Since not all joints are effected by the singularity in the same way, the result is that the robot will deviate from its calculated Cartesian path.

- *Control bandwidth*: the major job of the robot's (feedback) controller is to adapt the (instantaneous) *control* setpoint for the motors whenever an error is found between the desired end-effector motion and the measure one. Close to singularities, the robot will loose control bandwidth, since a similar Cartesian error gives rise to a tremendously larger joint error, at least for some joints. And large joint errors give rise to large actuator signals, but since most joint controllers are physically limited by joint actuator *saturation*, path tracking errors will be much more likely than far away from singularities.

  However, actuator saturation is not the only control problem that shows up close to singularities: since a singularity is a property of the *mechanism* that transforms motion between joint space and Cartesian space, the dynamical behaviour in Cartesian space will change drastically around singularities, without necessarily large changes in the dynamics in joint space. Hence, an appropriate change is required in the control gains for any Cartesian space control algorithm. Many robot controllers do not do this, which makes the robot control rather instable around singularities. Or, alternatively, the control gains are set to conservative values that keep the robot stable close to singularities, but then it becomes slower than necessary far away from singularities.

  A general solution to this control problem is as follows: the controller adapts its Cartesian *trajectory setpoints* close to singularities, but keeps it joint space controller gains. Section 5.10 explains the popular *damped-least squares* algorithm to support this approach.

- *Accuracy*: it's not only the *speed* with which the robot can suppress Cartesian errors that suffers from singularities, but also the motion accuracy. Since the chain looses at least one Cartesian motion degree of freedom, it just cannot execute any desired Cartesian motion accurately.

- *Load carrying capacity*: for kinematic chains that gain an uncontrollable passive degree of freedom, any part of the load carried by the end-effector that generates a reaction force in the direction of this uncontrollable degree of freedom cannot be stopped by the robot's motors. On the other hand, most kinematic chains (such as all serial industrial robot arms) become "infinitely stiff" in the direction of the motion degree of freedom that has been lost, since, by definition, the robot cannot move in this direction. This is sometimes a very positive effect, since the robot then needs not motor force to withstand any load in the singular direction (within the limits of plastic deformation of the chain's mechanical structure). This situation occurs, for

example, in the human arm, when it is fully stretched out: we cannot move any further in the stretched direction, but at the same time that is a comfortable, "bracing" posture to push heavy things forward since our muscles need not do any effort to withstand the pushing force.

All of these effects need to be coped with; one often made mistake is to try to solve everything in the control algorithm, but that most often misses relevant information about the goal of the current task, etc.

5.8.2 Numerical singularity detection via SVD

The *general* approach to find the singularities of a serial or tree-structured kinematic chain is via the *numerical* calculation the rank of the Jacobian. For a *square* Jacobian, $\det(\boldsymbol{J}) = 0$ is a necessary and sufficient condition for a singularity to appear. However, some chains do not have square Jacobians. Hence, a better numerical criterion is required. The most numerically stable criterion that finds the changes in the rank of the Jacobian is based on the *Singular Value Decomposition* (SVD), [85, 155, 247], that works for all possible kinematic structures: *every* matrix $\boldsymbol{A}$ (with arbitrary dimensions $m \times n$) has an SVD decomposition of the form

$$\underset{m\times n}{\boldsymbol{A}} = \underset{m\times m}{\boldsymbol{U}}\ \underset{m\times n}{\boldsymbol{\Sigma}}\ \underset{n\times n}{\boldsymbol{V}^T}, \quad \text{with} \quad \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \sigma_m & 0 & \dots & 0 \end{pmatrix}, \tag{5.32}$$

represented here for $n > m$. $\boldsymbol{U}$ and $\boldsymbol{V}$ are *orthogonal matrices*, and the singular values σ_i are in descending order: $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \geq 0$. $\boldsymbol{A}$ 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. This requires a computational cost of $\mathcal{O}(n^3)$. The *pseudo-inverse*, [166, 199, 141], $\boldsymbol{A}^\dagger$ of $\boldsymbol{A}$ is easily found as

$$\boldsymbol{A}^\dagger = \boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^T = \sum_{i=1}^{m} \frac{1}{\sigma_i}\boldsymbol{u}_i\boldsymbol{v}_i^T, \tag{5.33}$$

with

$$\boldsymbol{\Sigma}^{-1} = \begin{pmatrix} \sigma_1^{-1} & 0 & \dots & 0 \\ 0 & \sigma_2^{-1} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & \sigma_m^{-1} \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}. \tag{5.34}$$

The SVD is a *numerical* technique, with, in general, *no physical interpretation*. In the particular case of this Section, *no decomposition* of a Jacobian matrix exists in which $\boldsymbol{U}$, $\boldsymbol{V}$ or σ_i have meaningful physical units or physical interpretations. Nevertheless, too many publications keep on showing up in the robotics literature that (most often implicitly) attach physical meaning to singular values...

There is not much more to say about singularities *in general*, since their origin and effects depend on the *particular* geometry of a given kinematic chain. Hence, kinematic singularities will be treated further in later Sections that deal with chains with particular geometric designs.

5.9 Redundancy

A serial chain with n (actuated) joints is called redundant if it is used to perform a *Cartesian* task that requires less than the n available *joint space* degrees of freedom. In other words, the same Cartesian task can be performed with several different joint motions.

For example, a classical six degrees-of-freedom serial robot is redundant for a task in which it has to follow the surface of a workpiece with a vertex-like tool (Fig. 5.10) 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

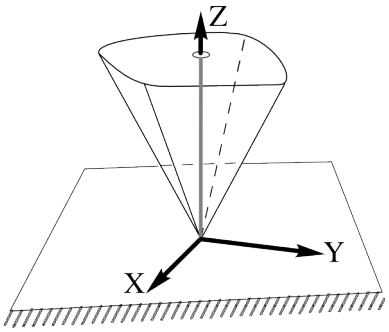

Figure 5.10: *Vertex-face* contact, as an example of a configuration of a tool that introduces redundancy in a robot task: the robot holding the tool can maintain this contact situation in many different configurations.

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.

One of the simplest examples of an *anthropomorphic* (i.e., " with human-like form") redundant robot is shown in Fig. 5.11. It has an extra joint between the "shoulder" and the "elbow" of the 6R wrist-partitioned manipulators in Figures 7.3 and 7.4. 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.

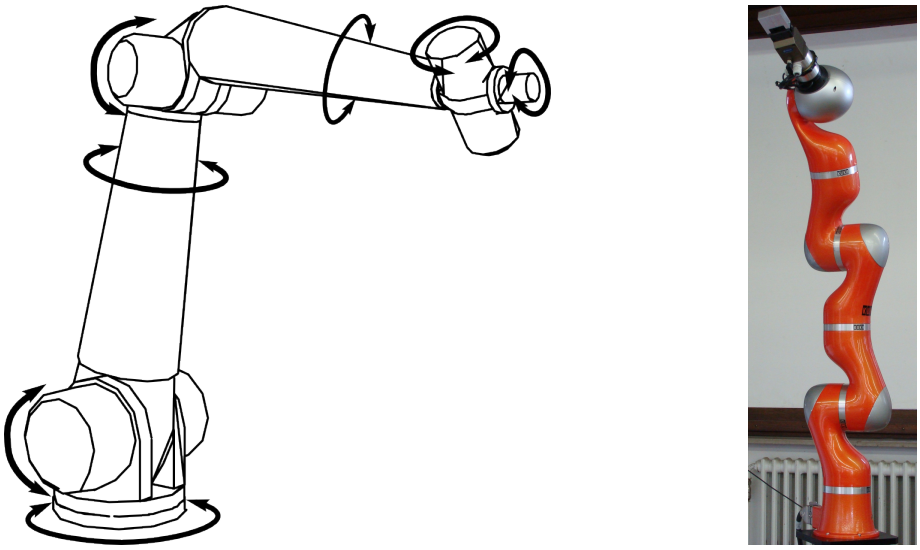

Figure 5.11: Redundant serial arms with seven revolute joints: three in the "shoulder" (i.e., the base of the robot), one in the "elbow", and three in the "wrist." This *313* configuration is the most popular extension of the typical six degrees of freedom industrial robots with the *321* design, by adding one revolute joint in the upper arm, just before the "elbow".

5.9.1 Redundancy resolution criteria

By definition, a redundant robot has more joint space motion freedom than required by its Cartesian space task, hence, 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 corresponding to the specified end-effector pose and twist. Some examples of redundancy resolution criteria are:

1. To keep the joints as close as possible to a specified configuration, in order to use the robot in the advantageous part of its workspace, or to avoid that joints reach their *mechanical limits*. A simple approach to

reach this goal is to attach *virtual springs* to the joints, whose equilibrium positions are near the middle of the desired motion range of the joints. With such a virtual spring model, the redundancy resolution criterion corresponds to the minimization of the potential energy in the springs.

2. To minimize the *kinetic energy* of the chain, [130], in order to put minimal overall load on the actuators. This approach requires the knowledge of the mass distribution of all segments (and also of the motor shafts, in the rather common case of electrical motors and high gear ratios).

3. To assign most of the power required for the motion to the most powerful motors in the robot, [200], or to the motors that are currently working closest to their most efficient range.

4. To 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, [70, 135, 176, 195]. This requires some quantitative "measure" of motion ability; it is well-known (but still not yet *widely* known!) that no such natural measures exist, [129, 151], so this criterion involves some arbitrary choices about how to weigh different components of the motion (more in particular, the linear and the rotational components).

5. To 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," [66, 115]. This criterion is very relevant for "bionic" robots (such as protheses and ortheses), or for computer models of the musculo-skeletal system.

6. To execute a high priority task while using the redundancy to achieve a lower priority task in parallel, [178]. In other words, the lower priority task can be executed (maybe only partially) in the *null space* of the high priority task.

7. To avoid obstacles in the robot's workspace. For example, a robot with an extra shoulder or elbow joint can "reach around" obstacles, [11, 113]. This approach is similar to the criterion of keeping the joints away from their limits; the difference is that the "springs" are now attached between the robot segments and the environment objects to avoid, and that they typically have a *repelling* behaviour.

8. To avoid singularities in the robot kinematics, [10, 13, 147, 189, 234]. For example, the 4R "Hamilton wrist," [149], (Fig. 5.12) can avoid the "wrist singularity" of Sect. 3.13. The addition of an extra joint is by itself not sufficient to have singularity-free motion, in the sense that even a redundant robot can end up in a singularity; it is the responsibility of the robot *controller* to generate a motion that *avoids* the singularity. The "internal spring" approach mentioned before is one possible way to automatically generate singularity-free motions in this case, but it is in general difficult to find the "obvious" locations to attach the virtual spring to, *and* this approach cannot *guarantee* singularity freedom at all times.

9. To travel through a singularity while keeping the joint velocities bounded, [23, 134].

10. To minimize the *RMS value* of the current through the motors, in order to avoid overheating and to reduce wear.

Some of these criteria optimize an *instantaneous* objective function (e.g., kinetic energy, or actuator power consumption), while others optimize a more *global* (i.e., non-instantaneous) objective function (e.g., closeness to a desired configuration, or minimal RMS actuator currents).

5.9.2 The extended Jacobian

Some of the redundancy resolution criteria of the previous section can be implemented via the concept of the *extended Jacobian*, [10, 136, 145]. This approach starts from the observation that the non-square $6 \times n$ Jacobian can be made into a square $n \times n$ matrix by extending it with $n - 6$ rows, collected in a $(n - 6) \times n$ matrix $\boldsymbol{A}$:

$$\widehat{\boldsymbol{J}} = \left(\begin{array}{c} \boldsymbol{J}(\boldsymbol{q}) \\ \hline \boldsymbol{A}(\boldsymbol{q}) \end{array} \right). \tag{5.35}$$

This is equivalent to adding $n - 6$ *linear constraints* on the joint velocities:

$$\boldsymbol{A}(\boldsymbol{q})\dot{\boldsymbol{q}} = 0. \tag{5.36}$$

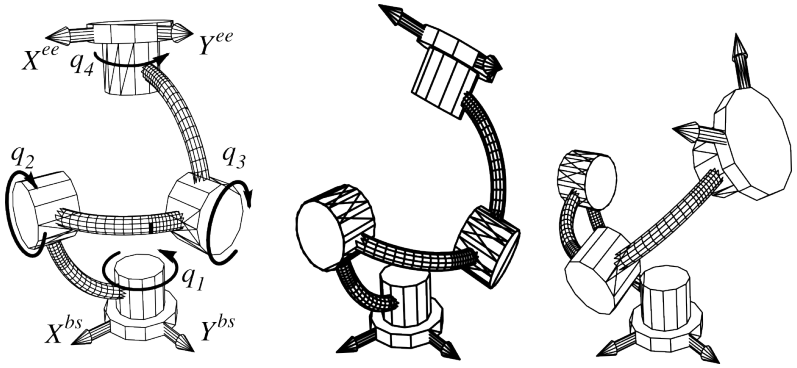

Figure 5.12: Redundant wrist with four revolute joints, in three different configurations. In the left-most configuration, two axes line up, but the wrist does not become singular.

In order to obtain a full-rank extended Jacobian $\widehat{\boldsymbol{J}}$, the constraint matrix $\boldsymbol{A}$ must be full rank by itself, and, in addition, be *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, [235]. Equation (5.35) then has a uniquely defined inverse:

$$\widehat{\boldsymbol{J}}^{-1} = \left(\boldsymbol{B} \mid *\right). \tag{5.37}$$

The $*$ indicates the part of the matrix $\widehat{\boldsymbol{J}}^{-1}$ which is irrelevant to the discussion in the following paragraphs.

5.9.3 Pseudo-inverse Jacobian: weighting in joint space

The relevant $n \times 6$ matrix $\boldsymbol{B}$ is a so-called *generalized inverse*, or *pseudo-inverse*, often denoted by $\boldsymbol{B} = \boldsymbol{J}^{\dagger}$, [18, 38, 164, 166, 167, 90]: it satisfies $\boldsymbol{J}\boldsymbol{B} = \mathbf{1}_{6\times 6}$ and $\boldsymbol{B}\boldsymbol{J} = \mathbf{1}_{n\times n}$. (This follows straightforwardly from the definition of $\widehat{\boldsymbol{J}}$.) With it, the forward velocity kinematics, Eq. (5.23), can be "inverted":

$$\dot{\boldsymbol{q}} = \boldsymbol{B}\ \mathbf{t}. \tag{5.38}$$

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. (5.38) in more detail, and in an alternative way for the particular example of the kinetic energy minimization criterion.

The oldest reference to the weighted pseudo-inverse approach is [275]; the oldest computationally efficient dynamic version seem to come from the Russian literature, [2, 207, 260, 261, 264].

The kinetic energy T of a serial manipulator is of the form

$$T = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\ \dot{\boldsymbol{q}}. \tag{5.39}$$

(TODO: prove it!) The $n \times n$ *joint space mass matrix* $\boldsymbol{M}$ (Sect. 6.5) is both invertible and symmetric, because T is a *positive scalar* and hence $T^T = T$. Minimizing the kinetic energy, while at the same time obeying the inverse kinematics requirement that $\mathbf{t} = \boldsymbol{J}\,\dot{\boldsymbol{q}}$, transforms the solution to the following *constrained optimization problem*:

$$\begin{cases} \min\limits_{\dot{\boldsymbol{q}}} \quad T &= \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\,\dot{\boldsymbol{q}}, \\ \text{such that} \quad \mathbf{t} &= \boldsymbol{J}(\boldsymbol{q})\,\dot{\boldsymbol{q}}. \end{cases} \tag{5.40}$$

The classical solution of this kind of problem uses *Lagrange multipliers*, [55, 247], i.e., the constraint in (5.40) is taken up into the functional T to be minimized as follows:

$$\min_{\dot{\boldsymbol{q}}} \quad T' = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{M}\,\dot{\boldsymbol{q}} + \boldsymbol{\lambda}^T\left(\mathbf{t} - \boldsymbol{J}\,\dot{\boldsymbol{q}}\right). \tag{5.41}$$

(For notational simplicity, the dependence of $\boldsymbol{M}$ and $\boldsymbol{J}$ on the n joint positions $\boldsymbol{q}$ is dropped.) $\boldsymbol{\lambda}$ is the 6×1 column vector of the (for the time being still unknown) *Lagrange multiplier* vector; physically, this can be interpreted

as the Cartesian *impulse vector* (Cartesian mass times Cartesian velocity), generated by violating the constraint $\mathbf{t} - \boldsymbol{J}\,\dot{\boldsymbol{q}} = 0$. (This violation can only happen if the chain does not have six independent Cartesian degrees of freedom!) The minimum of Eq. (5.41) with respect to $\dot{\boldsymbol{q}}$ is found by setting to zero the partial derivatives of the functional T', which determines the Lagrange multipliers $\boldsymbol{\lambda}$ as a function of the desired joint velocities:

$$\underset{1\times n}{\dot{\boldsymbol{q}}^{\,T}}\;\underset{n\times n}{\boldsymbol{M}} - \underset{1\times 6}{\boldsymbol{\lambda}^{T}}\;\underset{6\times n}{\boldsymbol{J}} = \mathbf{0}_{1\times n}. \tag{5.42}$$

This gives a set of n equations, in the n joint velocities and the six Lagrange multipliers. These Lagrange multipliers can be solved for by post-multiplying Eq. (5.42) by $\boldsymbol{M}^{-1}\boldsymbol{J}^{T}$:

$$\dot{\boldsymbol{q}}^{T}\boldsymbol{J}^{T} = \boldsymbol{\lambda}^{T}\left(\boldsymbol{J}\,\boldsymbol{M}^{-1}\boldsymbol{J}^{T}\right). \tag{5.43}$$

The left-hand side of this equation equals the transpose of the end effector twist, $\mathbf{t}^{T}$, 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. 5.8). Hence,

$$\boldsymbol{\lambda}^{T} = \mathbf{t}^{T}\left(\boldsymbol{J}\,\boldsymbol{M}^{-1}\boldsymbol{J}^{T}\right)^{-1}. \tag{5.44}$$

Equations (5.42) and (5.44), and the fact that $\boldsymbol{M}$ is symmetric, yield

$$\dot{\boldsymbol{q}} = \boldsymbol{M}^{-1}\boldsymbol{J}^{T}\left(\boldsymbol{J}\,\boldsymbol{M}^{-1}\boldsymbol{J}^{T}\right)^{-1}\mathbf{t} \tag{5.45}$$

$$= \boldsymbol{J}^{\dagger}_{M^{-1}}\mathbf{t}. \tag{5.46}$$

$\boldsymbol{J}^{\dagger}_{M^{-1}}$ 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*, [18, 38, 164]. It is rather straightforward to use an SVD-based approach to computing the weighted pseudo-inverse: just replace $\boldsymbol{A}$ in Eq. (5.32) by $\sqrt{\boldsymbol{M}}J$. Since $\boldsymbol{M}$ is a positive-definite matrix, it always has a "square root", although that square root is not uniquely defined.

It is not a good idea to calculate the solution $\dot{\boldsymbol{q}}$ of Eq. (5.45) by the straightforward matrix multiplications of Eq. (5.45); better numerical techniques exist, see e.g. [85] for a general matrix approach, and Chap. 6 for algorithms that exploit the kinematic structure of the manipulator.

5.9.4 Null space

While a *singularity* of a kinematic chain is mathematically detectable because the chain's Jacobian matrix looses rank (i.e., it is not of full *row* rank, Sect. 5.8), the mathematical way to detect *redundancy* is that the Jacobian matrix is not of full *column* rank: the different columns (each representing the motion generated by one joint of the chain) are linearly dependent, which means that some end-effector velocities can be generated by more than one set of joint velocities. In other words, the chain has a non-vanishing *null space*, i.e., a set of joint velocities $\dot{\boldsymbol{q}}^{N}$ that result in a "null velocity" of the end-effector(s):

$$\mathrm{Null}\left(\boldsymbol{J}(\boldsymbol{q})\right) = \left\{\dot{\boldsymbol{q}}^{N} \mid \boldsymbol{J}(\boldsymbol{q})\,\dot{\boldsymbol{q}}^{N} = \mathbf{0}\right\}. \tag{5.47}$$

Because of the Jacobian matrix, this null space is configuration dependent, i.e., it depends on the actual joint positions. Equation (5.47) implies that an arbitrary vector of the null space of the Jacobian can be used as an *internal motion* of the robot:

$$\mathbf{t}^{ee} = \boldsymbol{J}(\boldsymbol{q})\,\dot{\boldsymbol{q}} = \boldsymbol{J}(\boldsymbol{q})\left(\dot{\boldsymbol{q}} + \dot{\boldsymbol{q}}^{N}\right). \tag{5.48}$$

In other words, the internal velocity joint vector $\dot{\boldsymbol{q}}^{N}$ only moves the segments of the robot, but not its end effector(s).

(TODO: computation of null space!)

5.9.5 Cyclicity

The redundancy resolution approaches based on the Jacobian matrix yield only *local* optimality. For example, one minimizes the instantaneous kinetic energy, *not* the energy consumption during the complete task. Hence, 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 *can*

result in different *joint* trajectories during each cycle, [12, 17, 48, 62, 137, 174, 235]. Whether or not the joint space trajectory is cyclic depends on the *integrability* of the constraint equation (5.36), Sect. 4.13.2.

It can be proven that using the mass matrix of the robot as weighting in the weighted pseudo-inverse approach gives a cyclic result. (TODO: reference!)

5.10 Singularity-robust redundancy: damped least squares in joint space

This Section explains the *damped least-squares* algorithm, [36, 141, 177, 268], to solve the singularity problem for redundant robots. This approach applies so-called "(Tikhonov) regularization" techniques, developed in non-robotics contexts: [144] introduced the "damping" terminology into the least-squares solution approach, and [111, 157, 253] developed very similar ideas, without knowing about each other's work. The basic idea is that, since joint velocities become high in the neighbourhood of a singularity, *and* many different joint velocities generate the required end-effector velocity because the robot is redundant, one can give preference to those joint velocities that have the smallest "magnitude" (Sect. 4.9):

$$\min_{\dot{q}} \left\{ ||\dot{\boldsymbol{X}} - \boldsymbol{J}\dot{\boldsymbol{q}}||_C^2 + ||\dot{\boldsymbol{q}}||_\Lambda^2 \right\}, \tag{5.49}$$

with $||\boldsymbol{Y}||_X^2 = \frac{1}{2}\boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{Y}$, and with $\boldsymbol{\Lambda}$ an (arbitrarily chosen) weighting between the different joint velocities. This equation says (and the name of the algorithm alludes to the same fact) that one looks for a solution to the inverse kinematics problem (as represented by the first term in the equation) while keeping the ($\boldsymbol{\Lambda}$-weighted) norm of the solution $\dot{\boldsymbol{q}}$ as small as possible (as represented by the second term in the equation). One often sees Eq. (5.49) written in the somewhat less general form

$$\min_{\dot{q}} \left\{ ||\dot{\boldsymbol{X}} - \boldsymbol{J}\dot{\boldsymbol{q}}||_C^2 + \lambda^2||\dot{\boldsymbol{q}}|| \right\}, \tag{5.50}$$

with the latter norm of $\dot{\boldsymbol{q}}$ being the "Euclidean" norm. However, the norms used in the equations above are necessarily *arbitrary*, Sect. 4.9, *and* they are defined in *different* spaces: the C-norm $||\cdot||_C$ is chosen in Cartesian space, the $\boldsymbol{\Lambda}$-norm $||\cdot||_{\boldsymbol{\Lambda}}$ is chosen in joint space, such as the joint space mass matrix $\boldsymbol{M}$ of Eq. (5.39). However, *if* the Cartesian norm is chosen to represent *energy*, the "damping" name of the presented algorithm becomes clear: $||\dot{\boldsymbol{q}}||_{\boldsymbol{\Lambda}}^2$ is proportional to $\dot{\boldsymbol{q}}^T\boldsymbol{\Lambda}\dot{\boldsymbol{q}}$, so, in order for this result to be an energy, $\boldsymbol{\Lambda}$ must map the rightmost $\dot{\boldsymbol{q}}$ onto a (joint) force, hence $\boldsymbol{\Lambda}$ has the units of damping. Equation (5.49) can be solved in a similar same way as in Sect. 5.9.3, by taking the derivative with respect to $\dot{\boldsymbol{q}}$ and setting to zero. This yields:

$$\dot{\boldsymbol{q}} = \left(\boldsymbol{J}^T\boldsymbol{C}^{-1}\boldsymbol{J} + \boldsymbol{\Lambda}\right)^{-1}\boldsymbol{J}^T\boldsymbol{C}^{-1}\ \mathbf{t} = \boldsymbol{J}_{\boldsymbol{\Lambda}}^{\dagger}\ \mathbf{t}. \tag{5.51}$$

In the case $\boldsymbol{\Lambda}$ is a multiple $\lambda\mathbf{1}$ of the unit matrix $\mathbf{1}$ (and, as in Sect. 5.9.3, we use the "square root" trick for $\boldsymbol{C}$), it is straightforward to check that the SVD of the Jacobian matrix (Sect. 5.8.2) can be used to provide a numerically stable way to compute the damped least-squares pseudo-inverse $\boldsymbol{J}_{\boldsymbol{\Lambda}}^{\dagger}$:

$$\boldsymbol{J}_{\boldsymbol{\Lambda}}^{\dagger} = \sum_{i=1}^{m} \frac{\sigma_i}{\sigma_i^2 + \lambda^2}\boldsymbol{u}_i\boldsymbol{v}_i^T. \tag{5.52}$$

Since the singular values have no physical meaning, the interpretation of $\sigma_1/(\sigma_i^2+\lambda^2)$ is hard. But, quantitatively speaking, "close" to a singularity, some σ_i tends to zero, so the non-zero λ prevents the pseudo-inverse to become singular; "far" from singularities, all σ_i are "large", and the damping will have no effect; similarly, the pseudo-inverse reduces to the normal pseudo-inverse when the damping factor λ tends to zero. The above-mentioned straightforward check goes as follows:

- Let the SVD of $\boldsymbol{J}$ be $\boldsymbol{J} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, with $\boldsymbol{\Sigma} = \text{diag}(\sigma_i), i = 1, \ldots, m$.
- Eq. (5.51) reduces to $\boldsymbol{J}^{\dagger} = (\boldsymbol{J}^T\boldsymbol{J} + \lambda^2\mathbf{1})^{-1}\boldsymbol{J}^T$.
- Filling in the SVD decomposition yields:
  $\boldsymbol{J}^{\dagger} = \left(\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T + \lambda^2\mathbf{1}\right)^{-1}\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T = \left(\boldsymbol{V}\text{diag}(\sigma_i^2 + \lambda^2)\boldsymbol{V}^T\right)^{-1}\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T = \boldsymbol{V}\text{diag}\left(\frac{\sigma_i}{\sigma_i^2+\lambda^2}\right)\boldsymbol{U}^T$,
  since $\boldsymbol{U}^T\boldsymbol{U} = \mathbf{1}$ and $\boldsymbol{V}^T\boldsymbol{V} = \mathbf{1}$ (of appropriate dimensions).

The paragraphs above made the simplification $\mathbf{\Lambda} = \lambda\mathbf{1}$, only to make computations easier, but this choice does not influence the generality of the reasoning. However, this simplification takes away some flexibility in choosing appropriate damping:

- The well-known *Levenberg-Marquardt* algorithm, [144, 157], makes the choice $\mathbf{\Lambda} = \lambda \text{diag}\left(\boldsymbol{J}^T\boldsymbol{J}\right)$, which means the damping is scaled according to the "*curvature*" of the Jacobian, as expressed in its *Hessian matrix* $\boldsymbol{J}^T\boldsymbol{J}$.
- Typically, not all joints are equally effected by a singularity. For example, the singularities in a spherical wrist, Sect. 3.13, hardly influence the motion of the joints not connected to the wrist. Hence, an homogeneous damping $\mathbf{\Lambda} = \lambda\mathbf{1}$ for all joints will most likely result in a rather conservative value for λ. In principle, this can be overcome by a full damping matrix $\mathbf{\Lambda}$, whose values should be adapted to the current singularity.

The damped least-squares algorithm *only damps joint velocities*, that is, it doesn't *avoid* the real cause of the problem, namely a too high *desired* velocity for the end-effector near a singular configuration. Since the joint space-Cartesian space dynamics of the kinematic chain can change drastically near such a singularity, it would be much better to adapt the Cartesian motion specification to this difficulty, and lower the desired motion, *and/or* to change the motion in order to steer the robot *around* the singularity.

5.11 Redundancy resolution with constraints

(TODO! describe how to deal with (in)equality constraints on the positions, velocities, accelerations, or forces on the joints or on (some) end-effector directions, [2, 44, 145])

5.12 Underactuated motion: weighting in the Cartesian space

Assume the manipulator has *less* than six joints, say $6 - n$. Hence, the Jacobian $\boldsymbol{J}$ is a $6 \times n$ matrix, and the manipulator is constrained to move on a $(6 - n)$-dimensional sub-manifold of the six-dimensional space of translations and rotations. 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 Sect. 5.9.3, in that the (unconstrained) objective kinetic energy function to be minimized is:

$$\min_{\dot{\boldsymbol{q}}} \quad T = \frac{1}{2}\left(\mathbf{t}^{ee} - \boldsymbol{J}\dot{\boldsymbol{q}}\right)^T \boldsymbol{M}\left(\mathbf{t}^{ee} - \boldsymbol{J}\dot{\boldsymbol{q}}\right), \tag{5.53}$$

with $\boldsymbol{M}$ a (full-rank) (*Cartesian space!*) mass matrix. (As in the case of the joint space inertia matrix of Sect. 5.9.3, the choice of this Cartesian inertia matrix is in most cases fully arbitrary!) 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 (of the virtual *Cartesian* mass) 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 $\boldsymbol{J}^T$ 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

$$\begin{aligned}\dot{\boldsymbol{q}} &= (\boldsymbol{J}^T\boldsymbol{M}\boldsymbol{J})^{-1}\boldsymbol{J}^T\boldsymbol{M}\mathbf{t}^{ee}, \\ &= \boldsymbol{J}^{\dagger}_M\mathbf{t}^{ee}.\end{aligned} \tag{5.54}$$

$\boldsymbol{J}^{\dagger}_M$ 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*. (And not on the *joint space* of the robot!) Pre-multiplying Eq. (5.54) with $\boldsymbol{J}$ proves that the executed twist $\mathbf{t}$ is a *projection* of the desired twist $\mathbf{t}^{ee}$:

$$\mathbf{t} = \boldsymbol{J}\dot{\boldsymbol{q}} = \boldsymbol{J}\boldsymbol{J}^{\dagger}_M\mathbf{t}^{ee} = \boldsymbol{P}\mathbf{t}^{ee}, \quad \text{with } \boldsymbol{P} = \boldsymbol{J}\boldsymbol{J}^{\dagger}_M. \tag{5.55}$$

It is straightforward to check that $\boldsymbol{P}$ indeed satisfies the projection operator property that $\boldsymbol{PP} = \boldsymbol{P}$. A set of linear constraints similar to Eq. (5.36) does not exist: all joint velocities are possible.

5.13 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, Sect. 2.6.2. $\boldsymbol{G}$ represents the Cartesian directions in which the contact can generate contact forces, so it imposes a set of linear constraints on the joint velocities as in Eq. (5.36):

$$(\boldsymbol{G}^T \boldsymbol{\Delta J})\dot{\boldsymbol{q}} = 0. \tag{5.56}$$

This suggests a procedure to "filter" a given end-effector twist $\mathbf{t}^{ee} \in \text{span}(\boldsymbol{J})$ into a twist $\mathbf{t}$ compatible with the constraint (i.e., reciprocal to $\boldsymbol{G}$), similar to the redundancy resolution in Sect. 5.9.1. Indeed, this "*kinetostatic filtering*" [69] can be formulated as the following constrained optimization problem:

$$\begin{cases} \min\limits_{\mathbf{t}} \quad T &= \dfrac{1}{2}(\mathbf{t}^{ee} - \mathbf{t})^T \boldsymbol{M}\ (\mathbf{t}^{ee} - \mathbf{t}) \\ \text{such that} \quad \boldsymbol{G}^T \boldsymbol{\Delta t} &= 0. \end{cases} \tag{5.57}$$

The solution of this optimization problem runs along similar lines as the redundancy resolution problems in Sect. 5.9: (i) include the constraint in the objective function by means of Lagrange multipliers; (ii) set the partial derivative with respect to $\mathbf{t}$ equal to zero; and (iii) solve for the Lagrange multipliers. This leads to the following weighted projection operation:

$$\mathbf{t} = \left(\mathbf{1} - \left((\boldsymbol{\Delta G})^T\right)^{\dagger}_{M^{-T}} (\boldsymbol{\Delta G})^T\right)\mathbf{t}^{ee}. \tag{5.58}$$

5.14 Control versus simulation

Kinematic chains are extensively used in basically two major application domains: (i) the motion and force *control* of robots and machine tools, and (ii) *multi-body dynamics* ("MBD") in which one *simulates* the kinematic and dynamic behaviour of mechanisms, such as car suspensions, drive trains in machines, landing gears of airplanes (Fig. 5.8), etc. Both domains use kinematic chains with different goals (and hence also often with different computational algorithms):

- *Control*: the goal is to steer the chain's actuated joints in a realtime control loop to make the end-effector follow a specified trajectory, and/or execute specified forces on its environment. Since the control must execute at typical rates of *1000 times per second*, and the *real* kinematics and dynamics parameters of the chain are never exactly known, it is inevitable that desired and actual motions and forces differ. Hence, it is rather useless to spend time on computing the position and velocity closure equations to a numerical accuracy which is better than the inevitable errors of the controller.

  Singularities are an issue that the controller must deal with, since it is certain that they *will* occur sooner or later in the robot's tasks. Indeed, one of the major reasons to use robots is that they have much larger workspaces than machine tools (at the cost of lower stiffness and accuracy). However, the robots' reachable workspace contains singularities, that the controller must cope with, [147, 256]: (i) by changing the specified motion a little bit, in order *to avoid reaching* the singularity, or (ii) by reducing the speed of the robot to zero when it reaches the singularity, in order *to avoid actuator saturation* at the singularity.

  Redundancies are also an issue that the controller must deal with, since some the tasks that the robot will have to execute during its lifetime will not require all degrees-of-freedom that the robot can provide.

- *Multi-Body Dynamics simulation*: the goal here is to simulate the behaviour of mechanisms as *accurately as possible*, because one wants *to analyse* that behaviour even before the mechanisms have been built. Especially *force loadings* and *frequency analyses* are relevant to be simulated at design time, since there is not much to be done about the noise, the comfort and the wear that vibrations will cause once the mechanisms have been implemented in hardware. Hence, all kinematic and dynamic parameters are "exact",

the time resolution is typically in the orders of several hundreds of Hertz (which requires a simulation time step of many thousands of Hertz), but there are no realtime computational constraints. So, one wants to compute the position and velocity closures much more accurately than in the control case, and one can reduce the simulation time step to any small value that is necessary to achieve the desired numerical simulation accuracy, and one can even step backwards in time if needed (e.g., to reach a higher accuracy on the times of impact between two parts of a mechanism). The other difference with the control case is that the ranges of motion of the degrees of freedom in the mechanisms are typically much smaller than for robots, so singularities can always *be avoided* in the design, without having to take them into account actively during operation. In addition, redundancies do not occur, since the mechanisms are designed to have only those degrees-of-freedom that their normal functionality requires: mechanisms typically serve only one single purpose, so they do not have to change "tasks" in the same way as robots.

5.15 Closed Loop (Inverse) Kinematics (CLIK)

For robot control purposes, typical kinematics computations are (i) performed each closed loop sample time of the robot controller, (ii) always approximate, for various reasons, and (iii) most often computed via iterative numerical algorithms. Hence, some publications, [14] [47] [232] [282], have suggested to profit maximally from this fact, by letting the iterations of the kinematics algorithms coincide with those of the controller. Or, in other words, to let the controller "close" the position and velocity constraints of the chain: there is not much difference between (i) controlling away the error between the actual position and velocity of the *real* chain's end-effector(s), and (ii) controlling away the position and velocity gaps between the chain's *virtual* "end-effectors" resulting from cutting the chain into a tree. Indeed, the difference between both controllers is that the latter adds a couple of virtual end-effectors to the control problem, and that those have only virtual dynamics since they exist only in the model of the chain's kinematics, and not in reality. The result can be a tremendous gain in computational load, while still guaranteeing sufficient accuracy. Despite these advantages, this techniques is not at all common practice.

(TODO: give more details and concrete examples!)

Chapter 6

Dynamics

This Chapter adds *dynamics* to the kinematics treatment in Chap. 5. In the context of (bio)mechanical systems, dynamics is the study of how the *forces* acting on points and bodies make these objects *move*. This Chapter describes how to model the dynamic properties of (i) a point mass, (ii) a single rigid body, and (iii) a kinematic chain of interconnected rigid bodies, and how to calculate these objects' time evolution under a given set of internal and external forces and/or desired motion specifications.

6.1 Overview

The approach to derive rigid body and chain dynamics is rather straightforward: one extends *Newton's law* for the dynamics of a *point mass* to rigid bodies, which are indeed nothing else but conglomerations of point masses that keep constant distances with respect to each other. However, the *interconnection* of rigid bodies by means of (prismatic, revolute, ...) *joints* gives rise to new physical properties that do not exist for one single point mass or one single rigid body. More in particular, the *topology* of the kinematic chain (serial, fully parallel, tree, graph, ...) determines to a large extent the minimal complexity of the computational algorithms that implement these physical properties.

6.1.1 Application domains

Most of the introductory material can be found in textbooks on classical physics and mechanics, e.g., [9, 53, 79, 84, 230, 294], since research on the dynamics of multiple interconnected rigid bodies has not exclusively been performed with only robotics in mind, but has progressed more or less independently in different research communities, each with its own emphasis:

1. *Serial robot arms*, Chap. 7. The emphasis here is on efficient, realtime algorithms, that can be used in the controllers of commercial industrial robot arms to make them move faster and more accurately. "Realtime" means that the robot control computer must be able to perform the algorithms 1000 or more times per second. The Russian school of robotics [207, 260] was more than a decade ahead of the Western world [153], with respect to efficient algorithms for computing the dynamics of serial robot arms.

2. *Parallel manipulators*, also called *Parallel Kinematics Machines* (PKMs), Chap. 8. These structures typically have light-weight moving parts, because the motors can all be attached to the fixed base of the robot. Hence, PKMs are in principle capable of faster motions, with a higher payload/arm weight ratio, such that the importance of on-line dynamics algorithms increases with respect to serial robot arms. However, also the complexity of the dynamics algorithms is increased, because PKMs have several *kinematic loops*.

3. *Mobile manipulators*, [], Chap. 10. Many modern *service robots* consist of a *mobile platform* (a robot device with multiple actuated wheels), on which a *torso* is mounted with one or two *redundant arms*, and a *head* with also multiple degrees of freedom. All these components typically have rather different dynamic and kinematic properties: the base moves much slower than the arms or the head, and all wheels in the mobile platform feel *non-holonomic constraints*, Sect. 4.13. It is most often assumed that the constraints on the various end-effectors are *independent*, that is, no (virtual or real) *kinematic loops* exist.

4. *Spacecraft control*, [117, 118]. This domain mainly investigates the dynamics of a satellite, which is, to first order, a single, free-floating rigid body, but with important practical disturbances: the conservation of momentum must be obeyed; the structural flexibilities in light-weight moving parts such as long-reach arms and solar panels; the dynamics of fluid propellants; etc. Motion speeds are typically an order of magnitude smaller than serial robots, but when computing spatial re-orientation maneuvers in space, the (non-holonomic) constraint of conservation of angular momentum, and the fact that the satellite has no "inertial" reference frame does play an important role,

5. *Simulation of multibody systems: cars, trucks, trains, ...*, [61, 104, 216, 279]. The major emphases in this *multi-body dynamics* ("MBD") domain are on: the complexity generated by the huge amount of interconnected bodies and motion constraints (including *kinematic loops*), on the non-linearities in many of these elements (such as springs, dampers, and friction), and on the development of symbolic preprocessing and numerical integration schemes to cope with the system's large dimensions and multiple constraints. The *speed* of computation is only of secondary importance, since the algorithms are used mainly for simulation, during the analysis and design phases.

6. *Humanoid and walking robots.* A humanoid or a walking robot consists of a number of serial subchains: legs, arms, and head, all connected to the same trunk, which in itself consists of several bodies ("vertebrae") connected in series. The resulting topology is a *tree*, that means that there is only one way to go from one segment of the robot to another one.

   Humanoid and walking robots typically have some constraints imposed by their interaction with the environment: the feet of the robot are in contact with the ground; the arms can grasp objects that are fixed in the environment; the limbs of the robot must move around obstacles; the trunk of the robot must remain as vertical as possible; etc. Some of these interactions with the environment result in closed kinematic loops, others result in constraints on the joint accelerations. In addition, the robot programmer wants some parts of the robot to follow specific trajectories that are *virtual constraints*, and not *physical constraints* as in the examples before. Both physical and virtual constraints make the dynamics algorithms a bit more complex, similarly to the multibody dynamics case mentioned above. However, the number and nature of the constraints that act on a humanoid robot typically remain much lower and much simpler than those in multibody dynamics, hence allowing realtime solutions. Realtime performance is needed in the *control loops* of these robots.

   When running, both feet of a humanoid robot are in the air, and there is no fixed support point. This *free-floating base* situation is equivalent to the satellite dynamics mentioned above.

   Humanoid and walking robots have a lot of *redundancy*, in the sense 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.

7. *Computer graphics, games and animation.* Here, the emphasis is on *realistically looking* interactions between different (rigid and soft) bodies, but much less on *absolute physical realism.* 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: it begins to pay more attention to the *physically realistic* simulation of human and other figures, and realtime performance is a key feature in interactive games.

8. *Biomechanics and gait analysis.* In these domains, the real human is the subject of study, so the "robotic" models of the above-mentioned domains are not sufficient. The most distinctive particularity is the inclusion of realistic models of the human *musculo-skeletal system*, including the very non-linear and still incompletely understood muscle activation dynamics.

It turns out that the *tree-structured* kinematics chains of humanoid robots are the most complex ones for which the dynamics can be solved in realtime. So, this document discusses the dynamics of serial robots and humanoid/walking robots at the same time, because the latter is an only rather simple extension of the former.

An important focus of this document are the algorithms that allow *realtime execution* in a robot controller, i.e., the calculations require less time than the real-world physics. In practice, this means mostly that only ideal kinematic chains are considered: rigid bodies interconnected via ideal joints. Introducing flexible joints and joint friction increases the computational costs, although realtime execution remains possible. Introducing flexible

segments, however, leads to computations that cannot be done in realtime, because these systems cannot be modelled anymore as finite-dimensional *lumped-parameter systems*. Indeed, *infinite-dimensional* descriptions are required, involving *partial differential equations* (PDEs), instead of the *ordinary differential equations* (ODEs) that show up in *finite-dimensional* systems; and PDEs are much more difficult to solve than ODEs.

6.1.2 Practicalities

All dynamics algorithms discussed in this text assume that the physical parameters of the robot are known: dimensions of segments, relative positions and orientations of connected parts, mass distributions of segments, joints and motors. In practice, it's not straightforward to find realistic values for all these parameters in a given robot. In addition, the text assumes *ideal systems*, i.e., perfectly rigid bodies, joints without backlash or with perfectly known flexibilities and friction.

The computational complexity of an algorithm is most often expressed as $\mathcal{O}(N^k)$ ("order N to the kth power"). In the case of robot dynamics, 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 mostly with the most computationally efficient case, where $k = 1$, i.e., so-called *linear-time* algorithms.

The dynamics of serial, parallel and humanoid robots, and of mobile platforms, are well within the power of modern computer hardware and software, because its linear time complexity is due to the absence of *loops* in its kinematic chain structure, and the fact that all segments can be considered to be ideal rigid bodies. The simulation of *real human structures* is more complex: the skeleton has a number of kinematic loops; the joints are not ideal revolute, gimbal or spherical joints; and the muscles and tendons add "parallel actuator loops" around the kinematic structure. In addition, the dynamics of the human *muscle activation* is much more non-linear than the dynamics of typical electric motors.

Finding algorithms to calculate the dynamics is much more important for *serial* and *humanoid* 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 segments, and all motors are in, or close to, the base, such that the contributions of the manipulator inertias themselves are limited. The reader interested in parallel and mobile robot dynamics is referred to the literature, e.g., [37, 51, 127, 211, 224, 225, 291].

This document takes into account the inertia of the robot segments as well as that of the motors. This motor inertia can be significant, especially for the high *gear ratios* (of the order of 50–200 in most industrial robots) between motor shaft and robot segment.

6.1.3 Forward, inverse and hybrid dynamics

The core of this Chapter are the following two algorithms: forward dynamics, 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 segment, find the resulting end-effector acceleration $\ddot{\boldsymbol{X}}$."*

(In many cases (a humanoid robot, for example), the user is typically interested in *more than one* "end-effector": the two hands, two feet, the head, etc.) As mentioned before in this book (Sect. 3.17), the notations "$\dot{\boldsymbol{X}}$" and "$\ddot{\boldsymbol{X}}$" for the six-dimensional coordinate vector of a rigid body velocity, respectively an acceleration, are a bit misleading: it is *not* the second-order time derivative of any six-dimensional representation of position and orientation. However, the literature often uses this notation for convenience.

The Forward Dynamics is used for *simulation*, and *to predict* the behaviour of the robot during control, i.e., the FD calculates what the robot does when specific joint torques are applied, *and* under the assumption that all physical parameters in the dynamics model are accurate. 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 distribution of each segment, find the vector of joint forces $\boldsymbol{\tau}$ required to generate the desired acceleration."*

The ID is needed for:

1. *Control*: if one wants the robot to follow a specified trajectory, one has to convert the desired motion along that trajectory into joint forces that will generate this motion.

2. *Motion planning and optimization*: 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, and with what maximum speed the trajectory could be executed.

3. *Motion analysis*: the ID is able to give estimates of the forces acting in the human muscles, when the motion and ground reaction force are captured during, for example, a gait analysis measurement session.

The *Hybrid Dynamics* (HD) algorithm solves the following problem:

> *"Given some joint motions and forces, some "end-effector" motions and forces, as well as the mass distribution of each segment, find the resulting motion of the complete kinematic chain."*

Hybrid Dynamics is the major algorithm for the *posture control* of humanoid robots and mobile manipulators: the task typically requires specifications (of *some components*) of the motion and force on the end-effectors, while the robot also wants to keep its posture close to its most "comfortable" configuration.

6.1.4 Local and global dynamics

The forward and inverse dynamics algorithms are both *local*: they take the *instantaneous* motion and forces into account, and generate only instantaneous outputs. This local approach of dynamics cannot solve all relevant problems; for example, it doesn't provide an answer to the questions of which *gait* (walking, running, jumping, etc.) to use in a particular motion task for a humanoid or walking robot, or how to re-orient a free-floating robot. These problems are called *global dynamics* because their solution must look at the evolution of the system over much more than just the following time instant. Typical solutions of the global dynamics are *cyclic gaits*, i.e., a particularly synchronized trajectory of all joints, that is repeated over time. For example, running or swimming. The concrete time function of the synchronized motion could be found from a dynamic optimization, but this problem is more complex than the instantaneous, local dynamics; this text does not treat this problem.

6.2 Segment-to-segment recursions: mass, force, momentum

This Section extends the *kinematic* segment-to-segment recursions of Sect. 5.4 (i.e., position, velocity, and acceleration) with the *dynamics* recursions of inertia, force, and momentum. In contrast with the kinematic case, all coordinates in this Section are expressed with respect to the *same frame*, in order not to clutter the formulas with frame transformation matrices. These coordinate transformations will be introduced in Sect. 6.4, when the complete forward and inverse dynamics algorithms are constructed from the segment-to-segment recursions of this Section and Sect. 5.4. The *kinematic* acceleration recursions in that latter Section deal with the calculation of the acceleration of segment $i+1$ if the position, velocity *and* acceleration of the previous segment i are *given*, together with the acceleration *generated* at the joint between both segments. The acceleration recursion in this Section deals with how the *inertia*-dependent acceleration is transmitted over a joint; that means that it assumes that the joint acceleration is not imposed, and that the acceleration starts from *standstill* of the segments. Both components of the acceleration have to be added to find the real physical acceleration of joints and segments, which will be done in Sect. 6.4.

Figure 6.1 shows the basic building block of every recursive algorithm: segment i of the robot is connected to segment $i+1$ through a (in this example, revolute) joint. Forces $\boldsymbol{F}_i$ and $\boldsymbol{F}_{i+1}$ act on both segments, and these forces are related to the segments' accelerations $\ddot{\boldsymbol{X}}_i$ and $\ddot{\boldsymbol{X}}_{i+1}$ through their inertia matrices $\boldsymbol{M}_i$ and $\boldsymbol{M}_{i+1}$, Sect. 4.12.6. In contrast with Sect. 5.4, the following sections will use the indices "1" and "2" instead of "i" and "$i+1$", just for the sake of reducing the space required to write down some long equations. This Section explains

- how much of the mass matrix of the distal segment is felt by the proximal segment.
- how much of the *inertial* acceleration of a proximal segment is transmitted to its distal segment.
- how much of the force acting on the distal segment is transmitted to the proximal segment.

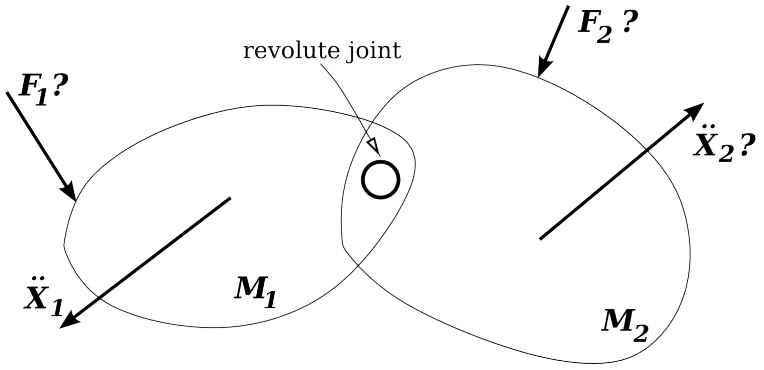

Figure 6.1: A rigid body is connected to another rigid body by an ideal revolute joint. That means that the joint constraint is such that: (i) it allows only rotational velocity and acceleration around its joint axis, and (ii) it cannot transmit a pure torque component about its axis. In this Figure, only the acceleration $\ddot{X}_1$ is assumed to be given.

6.2.1 Outward inertial joint acceleration recursion

The relationship between the acceleration $\ddot{\boldsymbol{X}}_1$ and the force $\boldsymbol{F}_1$ of segment 1 for an unconstrained segment 1 is given by the segment's mass matrix $\boldsymbol{M}_1$, Sect. 4.12.1. However, if segment 1 is connected to segment 2 by means of an ideal, *non-actuated* joint, the force $\boldsymbol{F}_1$ is not completely available to accelerate segment 1 because it has also to accelerate segment 2, through the connecting joint constraint. This section explains how to find the so-called *articulated inertia* $\boldsymbol{M}_1^a$ of segment 1 together with the segments it is connected to, i.e., the mapping from the acceleration of the segment to the corresponding force, taking into account the influence of the distal segment.

So, assume that segment 1 is *given* an acceleration $\ddot{\boldsymbol{X}}_1$ (Fig. 6.1), from an initially motionless configuration. In order to execute this acceleration, a certain force $\boldsymbol{F}_1$ is needed, and the goal of this Section is to find this force, and, at the same time, the articulated inertia $\boldsymbol{M}_1^a$ which is the ratio between force $\boldsymbol{F}_1$ and acceleration $\ddot{\boldsymbol{X}}_1$. Side-products of the derivation will be the force $\boldsymbol{F}_2$ that is transmitted through the mechanics of the joint and that is available to accelerate segment 2, and the corresponding acceleration $\ddot{\boldsymbol{X}}_2$. Both accelerations $\ddot{\boldsymbol{X}}_1$ and $\ddot{\boldsymbol{X}}_2$ can only differ in their component about the common joint axis:

$$\ddot{\boldsymbol{X}}_2 - \ddot{\boldsymbol{X}}_1 = \boldsymbol{Z}\ddot{q}_2, \tag{6.1}$$

with $\boldsymbol{Z}$ the six-dimensional basis vector of the joint (known from the position recursion), and $\ddot{q}$ the (as yet unknown) acceleration of the joint. In a frame with its Z axis on the joint axis, *and* for the considered case of a revolute joint, *and* for the chosen convention to let the angular velocity vector be represented by the first three numbers in the coordinate representation of $\boldsymbol{Z}$, that $\boldsymbol{Z}$ has the simple coordinate representation $(0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0)^T$. (This holds for the *particular choice* of twist coordinate representation $\mathbf{t}$ made in this document, Sect. 4.2.2: $\mathbf{t} = (\boldsymbol{\omega}\, \boldsymbol{v}_0)^T$.) The positive sign of q (and hence also of $\dot{q}$ and $\ddot{q}$) is *chosen* to be such that the relative motion between body 1 and body 2 has a positive component about the $\boldsymbol{Z}$ axis; in other words, $\dot{q} > 0 \Leftrightarrow (\dot{\boldsymbol{X}}_2)_Z > (\dot{\boldsymbol{X}}_1)_Z$. The (as yet unknown) transmitted force $\boldsymbol{F}_2$—represented by the *particular choice* $(\boldsymbol{f}, \boldsymbol{m})$ for the coordinate representation, Sect. 4.3—cannot have a component about the revolute joint axis, hence:

$$\boldsymbol{Z}^T \boldsymbol{\Delta} \boldsymbol{F}_2 = 0, \tag{6.2}$$

with $\boldsymbol{\Delta}$ as in Eq. (2.30), and repeated here for convenience:

$$\boldsymbol{\Delta} = \begin{pmatrix} \boldsymbol{0}_3 & \boldsymbol{I}_3 \\ \boldsymbol{I}_3 & \boldsymbol{0}_3 \end{pmatrix}. \tag{6.3}$$

This $\boldsymbol{\Delta}$ operator appears solely because of the above-mentioned *particular choices* of coordinate representations for velocity and force. When choosing, for example, an alternative representation for forces, $\boldsymbol{F} = (\boldsymbol{m}, \boldsymbol{f})$ instead of the $\boldsymbol{F} = (\boldsymbol{f}, \boldsymbol{m})$ chosen in this document, Eq. (6.2) simplifies to $\boldsymbol{Z}^T \boldsymbol{F}_2 = 0$; the $\boldsymbol{\Delta}$ would then also disappear from the equations below. The transmitted force $\boldsymbol{F}_2$ is fully used to accelerate the second body:

$$\boldsymbol{F}_2 = \boldsymbol{\Delta} \boldsymbol{M}_2 \ddot{\boldsymbol{X}}_2, \tag{6.4}$$

with $\boldsymbol{M}_2$ the mass matrix of segment 2. Equations (6.1)–(6.4), and the property $\boldsymbol{\Delta}\boldsymbol{\Delta} = \mathbf{1}$, yield:

$$-\boldsymbol{Z}^T\boldsymbol{M}_2\ddot{\boldsymbol{X}}_1 = \boldsymbol{Z}^T\boldsymbol{M}_2\boldsymbol{Z}\ddot{q}_2, \tag{6.5}$$

and

$$\ddot{q}_2 = -\left(\boldsymbol{Z}^T\boldsymbol{M}_2\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^T\boldsymbol{M}_2\ddot{\boldsymbol{X}}_1. \tag{6.6}$$

All factors on the right-hand side of this equation are known (the constant mass matrix $\boldsymbol{M}_2$ and joint axis $\boldsymbol{Z}$), or from a previous step in the recursion (i.e., $\ddot{\boldsymbol{X}}_1$). So, the joint acceleration $\ddot{q}_2$ corresponding to the proximal segment acceleration $\ddot{\boldsymbol{X}}_1$ is now known.

6.2.2 Outward inertial segment acceleration recursion

The segment acceleration $\ddot{\boldsymbol{X}}_2$ transmitted through the joint from proximal to distal segment, and caused by the segment acceleration $\ddot{\boldsymbol{X}}_1$, follows from Eqs (6.1) and (6.6):

$$\ddot{\boldsymbol{X}}_2 = \ddot{\boldsymbol{X}}_1 + \boldsymbol{Z}\ddot{q}_2, \tag{6.7}$$

$$= \left(\mathbf{1} - \boldsymbol{Z}\left(\boldsymbol{Z}^T\boldsymbol{M}_2\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^T\boldsymbol{M}_2\right)\ddot{\boldsymbol{X}}_1, \tag{6.8}$$

$$= \boldsymbol{P}_2^T\ddot{\boldsymbol{X}}_1. \tag{6.9}$$

$$\text{with } \boldsymbol{P}_2 = \mathbf{1} - \boldsymbol{M}_2\boldsymbol{Z}\left(\boldsymbol{Z}^T\boldsymbol{M}_2\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^T. \tag{6.10}$$

The last relationship follows from the symmetry of the mass matrix. The notation for the 6×6 matrix $\boldsymbol{P}_2$ is inspired from it being a *projection operator*, because projecting for a second time the result of a first projection does not change the result:

$$\boldsymbol{P}_2\boldsymbol{P}_2 = \boldsymbol{P}_2. \tag{6.11}$$

The projection $\ddot{\boldsymbol{X}}_2$ of the first segment acceleration $\ddot{\boldsymbol{X}}_1$ onto the second segment is not "orthogonal," but skewed by the mass distribution $\boldsymbol{M}_2$ of the second segment.

6.2.3 Outward force transmission

The force $\boldsymbol{F}_2$ transmitted through the joint from segment 1 to segment 2, as the result of an imposed acceleration $\ddot{\boldsymbol{X}}_1$ of segment 1, follows from Eqs (6.4) and (6.9):

$$\boldsymbol{F}_2 = \boldsymbol{\Delta}\boldsymbol{M}_2\ddot{\boldsymbol{X}}_2 = \boldsymbol{\Delta}\boldsymbol{M}_2\boldsymbol{P}_2^T\ddot{\boldsymbol{X}}_1. \tag{6.12}$$

$\boldsymbol{\Delta}\boldsymbol{M}_2\boldsymbol{P}_2^T$ doesn't have a projector operator interpretation, as it is easy to check algebraically that the projection operator condition (cf Eq. (6.11)) is not satisfied. But this should also be clear from physical reasoning: $\boldsymbol{\Delta}\boldsymbol{M}_2\boldsymbol{P}_2^T$ maps an acceleration into a force, and since these two entities live in physically distinct spaces, the corresponding operator cannot be a projection (since the latter must always be defined *within* one single space).

6.2.4 Inward mass matrix recursion

The force $\boldsymbol{F}_1$ needed to accelerate segment 1 by an amount $\ddot{\boldsymbol{X}}_1$ is the sum of (i) the inertial force to accelerate the first segment, and (ii) the force transmitted through the joint to accelerate the next segment:

$$\boldsymbol{F}_1 = \boldsymbol{\Delta}\boldsymbol{M}_1\ddot{\boldsymbol{X}}_1 + \boldsymbol{F}_2 \tag{6.13}$$

$$= \boldsymbol{\Delta}\boldsymbol{M}_1\ddot{\boldsymbol{X}}_1 + \boldsymbol{\Delta}\boldsymbol{M}_2\boldsymbol{P}_2^T\ddot{\boldsymbol{X}}_1, \tag{6.14}$$

$$= \boldsymbol{\Delta}\boldsymbol{M}_1^a\ddot{\boldsymbol{X}}_1, \tag{6.15}$$

$$\text{with} \quad \boldsymbol{M}_1^a = \boldsymbol{M}_1 + \left(\boldsymbol{I} - \boldsymbol{M}_2\boldsymbol{Z}\left(\boldsymbol{Z}^T\boldsymbol{M}_2\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^T\right)\boldsymbol{M}_2 = \boldsymbol{M}_1 + \boldsymbol{P}_2\boldsymbol{M}_2. \tag{6.16}$$

$\boldsymbol{M}_1^a$ the so-called *articulated body inertia*, [76], i.e., the increased inertia of segment 1 due to the fact that it is connected to segment 2 through an "articulation", that is, the revolute joint in this case. $\boldsymbol{M}_1^a$ of segment 1 is the sum of its own segment inertia $\boldsymbol{M}_1$ and the projected part $\boldsymbol{P}_2\boldsymbol{M}_2$ of the inertia of the second segment. It is

straightforward to check that $\boldsymbol{P}_2\boldsymbol{M}_2$ (and hence $\boldsymbol{M}_1^a$) is a symmetric matrix, as expected from an inertia matrix. This symmetry is analytically made clearer by the following equality, which is also straightforward to check:

$$\boldsymbol{P}_2\boldsymbol{M}_2 = \boldsymbol{P}_2\boldsymbol{M}_2\boldsymbol{P}_2^T. \tag{6.17}$$

The formula for the articulated inertia $\boldsymbol{M}_1^a$ in Eq. (6.10) looks very complicated at first sight, but in practice it is very efficient to calculate. Indeed, in the case that the joint is revolute, one can make sure that the Z-axis of the coordinate reference frame lies along the joint axis, and, hence, the 6×1 vector $\boldsymbol{Z}$ is equal to $(0\,0\,1\,0\,0\,0)^T$. So, $m_{66} = \boldsymbol{Z}^T\boldsymbol{M}_2\boldsymbol{Z}$ in Eq. (6.10) is the *scalar* element in the lower-right entry of the 6×6 matrix $\boldsymbol{M}_2$, and the inverse in this case is just the inverse of one single number. (In the more general case of n_i degrees of freedom in joint i, the matrix to be inverted is $n_i \times n_i$.) $\boldsymbol{P}_2$ in this particular frame becomes:

$$\boldsymbol{P}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & -m_{16}/m_{66} \\ 0 & 1 & 0 & 0 & 0 & -m_{26}/m_{66} \\ 0 & 0 & 1 & 0 & 0 & -m_{36}/m_{66} \\ 0 & 0 & 0 & 1 & 0 & -m_{46}/m_{66} \\ 0 & 0 & 0 & 0 & 1 & -m_{56}/m_{66} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \quad \text{(Wrong!!)} \tag{6.18}$$

Hence, the projection operator eliminates segment 2's component of inertia about the joint axis, i.e., the projected part of $\boldsymbol{M}_2$ is:

$$\boldsymbol{P}_2\boldsymbol{M}_2 = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} & m_{15} & 0 \\ m_{21} & m_{22} & m_{23} & m_{24} & m_{25} & 0 \\ m_{31} & m_{32} & m_{33} & m_{34} & m_{35} & 0 \\ m_{41} & m_{42} & m_{43} & m_{44} & m_{45} & 0 \\ m_{51} & m_{52} & m_{53} & m_{54} & m_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \quad \text{(Wrong!!)} \tag{6.19}$$

So, in case of a joint with a single degree of freedom, one needs a minima amount of computations to find $\boldsymbol{P}_2\boldsymbol{M}_2$!

6.2.5 Multi degrees-of-freedom joints

The paragraphs above assume that $\boldsymbol{Z}$ is a 6×1 vector, representing the axis of a 1 degree-of-freedom joint. However, this can easily be extended to N degrees-of-freedom joints: $\boldsymbol{Z}$ then becomes a $6 \times N$ matrix, and each of its columns represents one degree of motion freedom. The only resulting complication is that the inverse of $\boldsymbol{Z}^T\boldsymbol{M}_2\boldsymbol{Z}$ is now a matrix inverse, and not anymore a scalar inverse. However:

- Practical joints have one, two or at most three degrees of freedom, so the matrix inverse remains quite efficient to calculate.
- The reference frame on the joint can again be chosen in such a way that $\boldsymbol{Z}$ has mostly zero entries.

(TODO: make explicit calculation of $\boldsymbol{P}_2\boldsymbol{M}_2$ in the case of a spherical joint.)

6.2.6 Joints with shaft inertia

The paragraphs above assume the joint to be massless by itself. However, if the joint is actuated by an electrical motor, the motor shaft does have non-zero inertia. In addition, any gearbox between the motor shaft and the joint increases the effect of the motor shaft inertia, by a factor equal to the *square* of the *gear ratio*. Let d_2 denote the inertia of the motor shaft, as felt by the joint; that is, after transmission through a gearbox. Then, the reasoning between Eq. (6.2) and (6.16) can be repeated, with the following changes. The force $\boldsymbol{F}_2$ is not fully available to accelerate $\boldsymbol{M}_2$, but should also accelerate the motor shaft:

$$\boldsymbol{F}_2 = \boldsymbol{\Delta}\boldsymbol{M}_2\ddot{\boldsymbol{X}}_2 - d_2\ddot{q}_2\boldsymbol{Z}. \tag{6.20}$$

The *articulated body inertia* $\boldsymbol{M}_1^a$ becomes

$$\boldsymbol{M}_1^a = \boldsymbol{M}_1 + \boldsymbol{M}_2 - \boldsymbol{M}_2\boldsymbol{Z}\left(d_2 + \boldsymbol{Z}^T\boldsymbol{M}_2\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^T\boldsymbol{M}_2, \tag{6.21}$$

and, hence, the projection operator $\boldsymbol{P}_2$ becomes:

$$\boldsymbol{P}_2 = \mathbf{1} - \boldsymbol{M}_2\boldsymbol{Z}\left(d_2 + \boldsymbol{Z}^T\boldsymbol{M}_2\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^T \tag{6.22}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & -m_{16}/(d_2+m_{66}) \\ 0 & 1 & 0 & 0 & 0 & -m_{26}/(d_2+m_{66}) \\ 0 & 0 & 1 & 0 & 0 & -m_{36}/(d_2+m_{66}) \\ 0 & 0 & 0 & 1 & 0 & -m_{46}/(d_2+m_{66}) \\ 0 & 0 & 0 & 0 & 1 & -m_{56}/(d_2+m_{66}) \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \quad \text{(Wrong!!)} \tag{6.23}$$

(TODO: make the explicit calculation of $\boldsymbol{P}_2\boldsymbol{M}_2$ in the joint inertia case above; treat the case where the motor axis is not aligned with the joint axis.)

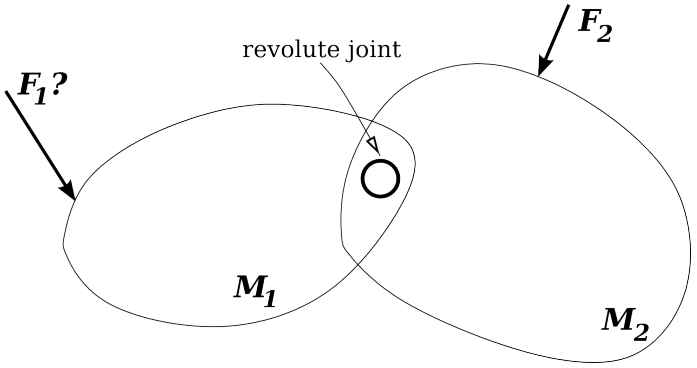

Figure 6.2: Similar construction as in Fig. 6.1, but now only the force $\boldsymbol{F}_2$ is given.

6.2.7 Inward force recursion

Assume now that a force $\boldsymbol{F}_2$ acts on segment 2, Fig. 6.2. (This force is *different* from the similarly named force $\boldsymbol{F}_2$ from Sect. 6.2.1, where the *acceleration* $\ddot{\boldsymbol{X}}_1$ was the given input.) The question is how much of this force is transmitted through the joint between segment 1 and 2. As in Section 6.2.4, segment 1 is assumed "fixed", in the sense that we are interested in the *force* as could be measured with a force sensor at segment 1, and not in the motion (i.e., acceleration) of segment 1 that results from that force.

The naive answer to the question of the transmitted force is to take the component $\boldsymbol{Z}^T\boldsymbol{\Delta}\boldsymbol{F}_2$ along the joint axis, and subtract it from $\boldsymbol{F}_2$. However, this answer is naive because it only considers the *static* force transmission and neglects the dynamics, that is, the inertial acceleration. So, although the *motion* caused by $\boldsymbol{F}_2$ can indeed only be around the joint axis, the resulting acceleration gives rise to inertial forces in *all* six spatial directions, because the mass matrix is in general not diagonal. So, the transmitted force is $\boldsymbol{F}_2$ minus all these acceleration-related components, and not just the component around the joint axis. The detailed derivation of this result goes as follows:

- $\boldsymbol{F}_2$ has a torque component $\tau = \boldsymbol{Z}^T\boldsymbol{\Delta}\boldsymbol{F}_2$ about the joint axis.
- $\boldsymbol{Z}^T\boldsymbol{M}_2\boldsymbol{Z}$ is the inertia of segment 2 about the joint axis.
- The dynamic equation $\boldsymbol{F} = \boldsymbol{\Delta}\boldsymbol{M}_2\ddot{\boldsymbol{X}}$ yields the following relationship between a *unit* torque about the joint axis ($\boldsymbol{F} = \boldsymbol{\Delta}\boldsymbol{Z}$) and the resulting acceleration $\boldsymbol{\Delta}\boldsymbol{M}_2\ddot{\boldsymbol{X}}$.
- Taking the component α of this acceleration about the joint axis ($\alpha = \boldsymbol{Z}^T\ddot{\boldsymbol{X}}$) yields $\alpha = \boldsymbol{Z}^T\boldsymbol{M}_2^{-1}\boldsymbol{\Delta}\boldsymbol{\Delta}\boldsymbol{Z} = (\boldsymbol{Z}^T\boldsymbol{M}_2\boldsymbol{Z})^{-1}$.
- $\alpha\tau$ is the magnitude of the total acceleration of segment 2 about the axis.
- This spatial acceleration $\boldsymbol{A}$ can only be around the joint axis $\boldsymbol{Z}$, hence $\boldsymbol{A} = \boldsymbol{Z}\alpha\tau = \boldsymbol{Z}(\boldsymbol{Z}^T\boldsymbol{M}_2\boldsymbol{Z})^{-1}\boldsymbol{Z}^T\boldsymbol{\Delta}\boldsymbol{F}_2$.
- This spatial acceleration generates a spatial force $\boldsymbol{\Delta}\boldsymbol{M}_2\boldsymbol{A} = \boldsymbol{\Delta}\boldsymbol{M}_2\boldsymbol{Z}(\boldsymbol{Z}^T\boldsymbol{M}_2\boldsymbol{Z})^{-1}\boldsymbol{Z}^T\boldsymbol{\Delta}\boldsymbol{F}_2$.

So, the part $\boldsymbol{F}_1$ of the force $\boldsymbol{F}_2$ transmitted in inward direction to segment 1 is then the original force $\boldsymbol{F}_2$ minus the latter spatial force which is "lost" in accelerating segment 2:

$$\boldsymbol{F}_1 = \boldsymbol{F}_2 - \boldsymbol{\Delta M}_2 \boldsymbol{Z} (\boldsymbol{Z}^T \boldsymbol{M}_2 \boldsymbol{Z})^{-1} \boldsymbol{Z}^T \boldsymbol{\Delta F}_2, \tag{6.24}$$

$$= \boldsymbol{P}_2^S \boldsymbol{F}_2, \tag{6.25}$$

with $\boldsymbol{P}_2^S$ the spatial transpose (Eq. 2.31) of the projection operator $\boldsymbol{P}_2$ (Eq. 6.10). The force recursion is *independent* of the accelerations $\ddot{\boldsymbol{X}}_1$ and $\ddot{\boldsymbol{X}}_2$: the force discussed in this Section *adds* to the force on the second segment caused by the acceleration of the first segment.

6.2.8 Momentum recursion

TODO, [183].

6.2.9 Inward articulated mass coordinates recursion

Section 6.2.4 explained the recursion of the articulated mass, but with all vectors expressed in the same (implicitly identified) reference frame. This Section adds explicitly the reference frame transformations between two proximal frames on neighbouring segments. Recall this book's convention (Sect. 5.3.4) to express all coordinates with respect to the proximal frames of segments, and that joint i is attached to the proximal frame $\{p_i\}$ of segment i.

The inward articulated mass matrix recursion calculates the articulated mass $\boldsymbol{M}_i^a$ felt by segment i, and with the origin of the proximal frame $\{p_i\}$ of the segment as reference point, when the articulated mass matrix $\boldsymbol{M}_{i+1}^a$ of the distal segment $i+1$ (with the origin of its own proximal frame as reference point), and the mass matrix $\boldsymbol{M}_i$ of segment i is also known in the ith proximal frame. Section 6.2.4 explained already how the mass matrix is transmitted through the revolute joint, Eq. (6.16). Hence, the coordinate form of the inward recursion $\boldsymbol{M}_i^a \leftarrow \boldsymbol{M}_{i+1}^a$ becomes:

$$\boldsymbol{M}_i^a = \boldsymbol{M}_i + {}^{i}_{i+1}\boldsymbol{S}^T \, \boldsymbol{P}_{i+1} \boldsymbol{M}_{i+1}^a \, {}^{i}_{i+1}\boldsymbol{S}, \tag{6.26}$$

because the projected articulated inertia $\boldsymbol{P}_{i+1}\boldsymbol{M}_{i+1}^a$ expressed in $\{p_{i+1}\}$ must be transformed to $\{p_i\}$, as in Eq. (4.79). $\boldsymbol{M}_i^a$ can be interpreted as an operator working on the acceleration of segment i, and resulting in a force, all expressed in the proximal frame $\{p_i\}$ of that segment. So, in Eq. (6.26), one recognizes the following steps from right to left:

1. ${}^{i}_{i+1}\boldsymbol{S}$ transforms the coordinates of the acceleration of segment i from proximal frame $\{p_i\}$ to proximal frame $\{p_{i+1}\}$.
2. There, that acceleration works on the projection $\boldsymbol{P}_{i+1}\boldsymbol{M}_{i+1}^a$ of the articulated mass matrix of segment $i+1$ onto segment i, and generates a force.
3. Finally, ${}^{i}_{i+1}\boldsymbol{S}^T$ transforms the coordinates of this force back to the proximal frame $\{p_i\}$ of segment i.

6.2.10 Inward force coordinates recursion

This section explains the recursion from $\boldsymbol{F}_{i+1}$, the total force felt at the origin of the proximal frame of segment $i+1$, to $\boldsymbol{F}_i$, the total force felt at the origin of the proximal frame of segment i. $\boldsymbol{F}_i$ consists of three parts:

1. *Contributions from segment $i+1$:* the accumulated resultant total force $\boldsymbol{F}_{i+1}$. The transmitted part of this force is given in Eq. (6.25), with the projection operator taking into account the joint shaft inertia, as in Eq. (6.22).
2. *Contributions from the joint torque τ_{i+1}*:

$$\boldsymbol{\Delta M}_{i+1}^a \boldsymbol{Z}_{i+1} (d_{i+1} + \boldsymbol{Z}_{i+1}^T \boldsymbol{M}_{i+1}^a \boldsymbol{Z}_{i+1})^{-1} \tau_{i+1}. \tag{6.27}$$

From right to left, the terms in this equation have the following physical meaning:

- the torque accelerates the inertia it feels around the joint axis, consisting of the shaft inertia d_{i+1} plus the component $\boldsymbol{Z}_{i+1}^T \boldsymbol{M}_{i+1}^a \boldsymbol{Z}_{i+1}$ of the articulated mass of segment $i+1$.
- the resulting joint acceleration causes a spatial acceleration along $\boldsymbol{Z}$.

- the resulting accelaration causes a spatial force via the inertia mapping $\boldsymbol{\Delta M}^a_{i+1}$.

3. *Contributions from segment i:*

   (a) The velocity-dependent bias force $\boldsymbol{F}^b_i$, generated by the angular velocity and the mass properties of segment i, Eq. (4.89).

   (b) The "external force" $\boldsymbol{F}^e_i$, i.e., the resultant of all forces applied to segment i, for example by people or objects pushing against it.

These components constitute almost all physical contributions, except the force generated by *accelerating* segment i: this component is added later, in specific dynamics algorithms, because depending on that algorithm, the acceleration of segment i comes either from a *user-specified* acceleration of segment i, or from the joint torque applied to joint i. In matrix form, the recursion $\boldsymbol{F}_i \leftarrow \boldsymbol{F}_{i+1}$ becomes:

$$\boldsymbol{F}_i = {}^{i+1}_{i}\boldsymbol{S}\left\{\boldsymbol{P}^S_{i+1}\boldsymbol{F}_{i+1} - \boldsymbol{\Delta M}^a_{i+1}\boldsymbol{Z}_{i+1}(d_{i+1} + \boldsymbol{Z}^T_{i+1}\boldsymbol{M}^a_{i+1}\boldsymbol{Z}_{i+1})^{-1}\tau_{i+1}\right\} + \boldsymbol{F}^b_i + \boldsymbol{F}^e_i. \tag{6.28}$$

The minus sign for the joint torque contribution comes from the fact that segment i feels a torque $-\tau_{i+1}$ if the motor at joint $i+1$ applies a torque of $+\tau_{i+1}$.

Again, the three different contributions to the segment force appear *linearly* in the equations, because the *superposition principle* holds: the effects of one contribution can be calculated separately, and added to the effects of the other contributions when required. For example, one can calculate the motion of the robot under gravity, without actuation at the joints, by just eliminating the joint torque contributions in the force recursion.

6.3 Recursions through tree-structured chains

The outward and inward recursions of the previous Sections are one joint at a time. These recursions apply directly to *serial* kinematic chains, and with only simple changes to *tree*-structured chains:

- Outward recursions (position, velocity and acceleration): no changes necessary, since every segment and joint is on only one serial path from the root of the tree.
- Inward recursion of articulated mass: when segment i is a branching node in a tree, Eq. (6.26) must be changed to contain the *sum* over all $\boldsymbol{M}^a_{i+1}$ of the distal segments connected to segment i.
- Inward recursion of force: when segment i is a branching node in a tree, Eq. (6.28) contains the *sum* over all $\boldsymbol{F}_{i+1}$ of the segments connected to segment i, plus the joint torque contributions of all joints at the distal frames $i+1$.

6.4 Forward and inverse dynamic algorithms

This Section combines the material of Sections 5.4, 6.2 and 6.3, to construct *linear-time* algorithms for the forward and inverse dynamics of serial and tree-structured kinematic chains. This Section still makes the assumption that the *root* of the kinematic chain is fixed to the ground; this constraint will be relaxed in Sect. 6.7.4. The relationship between the force τ_i delivered at a joint and the acceleration $\ddot{q}_i$ of that joint is given by:

$$\tau_i = (d_i + \boldsymbol{Z}^T_i\boldsymbol{M}^a_i\boldsymbol{Z}_i)\ddot{q}_i + \boldsymbol{Z}^T_i(\boldsymbol{F}_i + \boldsymbol{M}^a_i\ddot{\boldsymbol{X}}_{i-1}). \tag{6.29}$$

The factor in front of the joint acceleration $\ddot{q}_i$ is the inertia felt on the joint axis, being the sum of the axis' own inertia d_i and the component on the joint axis of the articulated inertia $\boldsymbol{M}^a_i$ of all segments more distal than i. The joint also feels (the component on the joint axis of) the external and Coriolis forces $\boldsymbol{F}_i$ from the distal segments, and the inertia force $\boldsymbol{M}^a_i\ddot{\boldsymbol{X}}_{i-1}$ of its own segment's motion, caused by the more proximal joints.

6.4.1 Inverse dynamics (ID) with given joint accelerations

Equation (6.29) *is* already the ID, and one just has to calculate all factors on the right-hand side of this equation. The acceleration $\ddot{\boldsymbol{X}}_N$ of each end-effector segment "N" is to be specified by the user. For the time being, assume that this desired end-effector acceleration has already been transformed into corresponding desired joint angle accelerations $\ddot{q}_i$ for each joint i. For robots with less or more than six joints, this transformation can be non-trivial and/or non-unique. The following calculations have to be performed:

1. *Outward motion recursion* of position and velocity and acceleration (including the desired acceleration as well as the bias acceleration due to the angular velocities). The result is that, for each segment, the position, velocity and acceleration are known with respect to the chain's root.

2. *Inward articulated mass matrix recursion.* Initialized by $\boldsymbol{M}_N^a = \boldsymbol{M}_N$ at each leaf segment of the tree (assumed to have the index "N"). The result is that each joint knows what inertia it feels from all the more distal segments.

3. *Inward force recursion.* Initialized by the given external forces $\boldsymbol{F}_N$ on each leaf segment. The result is that each joint knows what force it feels from all the more distal segments.

4. The *outward joint torque recursion* finds the joint torques needed to generate the desired accelerations, given the articulated inertias felt at each joint, from Eq. (6.29). (Note that no real *recursion* is needed, since all joint accelerations have already been given, so the spatial accelerations of all segments have been calculated in the first step of the algorithm already.)

6.4.2 Forward dynamics (FD)

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, except that the joint accelerations are not yet known.

2. *Inward articulated mass matrix recursion.* Same as for ID.

3. *Inward force recursion.* Same as for ID, except that no joint accelerations are known yet.

4. *Outward acceleration recursion.* The acceleration at joint i follows straightforwardly from Eq. (6.29):

$$\ddot{q}_i = (d_i + \boldsymbol{Z}_i^T \boldsymbol{M}_{i+1}^a \boldsymbol{Z}_i)^{-1} \left\{ \tau_i - \boldsymbol{Z}_i^T \left(\boldsymbol{F}_{i+1} + \boldsymbol{M}_{i+1}^a \ddot{\boldsymbol{X}}_i \right) \right\}. \tag{6.30}$$

   Gravity is taken into account by initializing the recursion with the gravitational acceleration: $\ddot{\boldsymbol{X}}_0 = \boldsymbol{g}$. Once $\ddot{q}_i$ is known, the total acceleration of the ith segment can be found:

$$\ddot{\boldsymbol{X}}_i = \ddot{\boldsymbol{X}}_{i-1} + \ddot{q}_i \boldsymbol{Z}_i + \ddot{\boldsymbol{X}}_{b,i}, \tag{6.31}$$

   with $\ddot{\boldsymbol{X}}_{b,i}$ the *bias acceleration* due to the non-vanishing angular velocities of the proximal segment, Eq. (5.22). Now, the joint acceleration calculation of the next joint $\ddot{q}_{i+1}$ can be done, etc.

5. *Integration of joint accelerations* into joint velocities and joint positions. (Numerical integration algorithms are not discussed in this text.)

6.4.3 Hybrid dynamics (HD)

The HD scheme is a straightforward variation on the Inverse and Forward Dynamics schemes: during the inward and outward recursions, the given accelerations and/or forces are filled in when the corresponding joints and segments are encountered.

6.5 Analytical form of tree structure 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:

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{c}(\dot{\boldsymbol{q}}, \boldsymbol{q}) + \boldsymbol{g}(\boldsymbol{q}). \tag{6.32}$$

The matrix $\boldsymbol{M}(\boldsymbol{q})$ is called the *joint space mass matrix*. $\boldsymbol{M}(\boldsymbol{q})$ gives the *linear* relationship between the joint forces and the resulting joint accelerations, *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

$$T = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}. \tag{6.33}$$

(TODO: prove it! Make the transformation from recursive form to analytical form explicitly!)

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. (6.32) becomes

$$\tau_i = \sum_j \boldsymbol{M}_{ij}(\boldsymbol{q})\,\ddot{q}_j + \sum_{j,k} \boldsymbol{C}_{ijk}(\boldsymbol{q})\,\dot{q}_j\dot{q}_k + \boldsymbol{g}_i(\boldsymbol{q}). \tag{6.34}$$

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.

6.6 Composite inertia method

Equation (6.34) gives rise to the oldest dynamical algorithm, the so-called *composite inertia* method:

- *Inverse dynamics.* The joint torques needed to accelerate the robot with given joint accelerations $\ddot{\boldsymbol{q}}$ are found immediately from Eq. (6.34). The computational effort is in calculating the joint space mass matrix, and the other right-hand side terms in that equation. The straightforward method to do so is the inward and outward sweep recursions presented before.

- *Forward dynamics.* Finding the joint accelerations that results from given joint torques are found from Eq. (6.34) by inverting the joint space mass matrix:

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}(\boldsymbol{q})\,\boldsymbol{\tau} - \boldsymbol{c}(\dot{\boldsymbol{q}}, \boldsymbol{q}) - \boldsymbol{g}(\boldsymbol{q}). \tag{6.35}$$

  This inversion of the joint space mass matrix has a computational complexity of order $\mathcal{O}(n^3)$, with n the number of joints.

The recursive method presented in previous Sections is linear in the number of joints, $\mathcal{O}(n)$, so it *seems* to be more efficient than the composite inertia method. However, the total number of calculations turns out to be less in the composite inertia method, for structures with a low number of joints, [**?**].

6.7 Acceleration constraints—Weighting approach

This Section, and Sect. 6.8, explain how to compute the forward and inverse dynamics when *acceleration (equality) constraints* are added to some segments on a kinematic chain. The constraints can be *physical* (e.g., contacts with the environment) or *virtual* (e.g., desired motion of (part of) a frame attached to a segment), [207, 262]. This Section, as well as all papers cited here, assume that the constraints on the various end-effectors are *independent*, that is, no (virtual or real) *kinematic loops* exist.

Only *acceleration* constraints are considered; velocity or position constraints can be transformed into acceleration constraints by simple derivation. However, derivation is known to introduce numerical complications, with respect to numerical accuracy of the computations, *and* it requires the position and velocity constraints to hold perfectly already at the start of the computation of the influence of the the acceleration constraint.

The differences between this Section and Sect. 6.8 are:

- this Section solves *forward* and *inverse dynamics*; Sect. 6.8 only the inverse dynamics.
- the acceleration constraints in this Section can be *partial*, while Sect. 6.8 works only with full six-dimensional accelerations.
- this Section *weighs* different constraints if they can not all be satisfied at the same time; Sect. 6.8 first solves the *highest priority* constraint, and then uses the *null space* of that constraint to solve the second highest priority, etc.

6.7.1 End-effector constraint in serial chain

First, assume only the *end-effector* segment, "N", in a *serial* chain is not allowed to accelerate arbitrarily, but must satisfy the following (linear) constraints:

$$\boldsymbol{A}_N^T \ddot{\boldsymbol{X}}_N = \boldsymbol{b}_N. \tag{6.36}$$

If there are m constraints, $\boldsymbol{A}_N$ is a $6 \times m$ matrix, and each of its columns can be interpreted as a *constraint force* acting on the segment. The right-hand side m-dimensional vector $\boldsymbol{b}_N$ represents the "*acceleration energy*" generated in the constraints. Classical mechanics has no generally accepted name for this product of a force with an acceleration, although Gauss [80], Hertz [108], Gibbs [81], and Appell [7, 8] already used it. (In differential geometry, $\boldsymbol{b}_N$ is the (square of the) *geodesic curvature* of the motion, [109].)

The constraints can be *physical*, e.g., the segment is attached to, or in contact with, objects in the environment that are physically constrained themselves. Or the constraints can be *virtual*, in the sense that the represent *desired acceleration* imposed on the segment by the human programmer; for example:

- to move a reference *point* on the segment vertically, the constraint matrices can be as follows:

$$\boldsymbol{A}_N = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \boldsymbol{b}_N = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{6.37}$$

  The columns of $\boldsymbol{A}_N$ are constraint forces in the horizontal X and Y directions, that must keep the acceleration in those directions to zero. (Recall the convention of this book (Sect. 4.3) to represent a force as $\boldsymbol{F} = (\boldsymbol{f}^T, \boldsymbol{m}^T)^T$.)

- to move the segment vertically without allowing rotations, the constraint matrices represent five constraints:

$$\boldsymbol{A}_N = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \boldsymbol{b}_N = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{6.38}$$

  That means that the constraining forces and moments are allowed to work in all directions, except the vertical Z direction.

- to give the segment a desired acceleration $\ddot{\boldsymbol{X}}_N$, the constraint matrices can be as follows:

$$\boldsymbol{A}_N = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \boldsymbol{b}_N = \ddot{\boldsymbol{X}}_N. \tag{6.39}$$

The m constraint forces $\boldsymbol{A}_N$ are not known in advance, but will be found by the dynamics algorithm below. More precisely, the *working directions* of all constraint forces are known (these are the "unit" matrices $\boldsymbol{A}_N$ from the examples above), but not their $m \times 1$ vector of *magnitudes* $\boldsymbol{\nu}$. So, the real constraint forces (at the end-effector) will be $\boldsymbol{A}_n boldsymbol\nu$. The introduction of segment constraints leads to extensions to the recursive formulas of Sect. 6.4:

- The constraint forces in the matrix $\boldsymbol{A}_N$ are propagated by the inward force recursion of Eq. (6.25):

$$\boldsymbol{A}_i = \boldsymbol{P}_{i+1}^S \boldsymbol{A}_{i+1}. \tag{6.40}$$

- During the inward recursion, we keep track of how much of the "constraint acceleration energy" $\boldsymbol{b}_N$ is already generated *by the (external and inertial) forces, and by the applied joint torques* (and, hence, need not be generated by the artificial constraints anymore):

$$\boldsymbol{\beta}_i = \boldsymbol{\beta}_{i+1} + \boldsymbol{A}_{i+1}^T \boldsymbol{\Delta} \left\{ \ddot{\boldsymbol{X}}_{i+1} + \boldsymbol{Z}_i D^{-1} \left(\tau_{i+1} - \boldsymbol{Z}_i^T (\boldsymbol{F}_{i+1} + \boldsymbol{M}_{i+1}^a \ddot{\boldsymbol{X}}_{i+1}) \right) \right\}, \tag{6.41}$$
$$\text{with} \quad D = d_{i+1} + \boldsymbol{Z}_i^T \boldsymbol{M}_{i+1}^a \boldsymbol{Z}_i, \quad \text{and} \quad \boldsymbol{\beta}_N = 0.$$

  The terms behind the $\boldsymbol{\Delta}$ operator represent all accelerations that result from inertial forces behind the joint, and from the force applied to the joint.

- The inward recursion keeps track of how much of the constraint acceleration energy $\boldsymbol{b}_N$ is already generated *by each of the virtual unit constraint forces* $\boldsymbol{A}_N$ themselves, via the $m \times m$ acceleration constraint coupling matrix $\boldsymbol{\mathcal{Z}}_i$:[1]

$$\boldsymbol{\mathcal{Z}}_i = \boldsymbol{\mathcal{Z}}_{i+1} - \boldsymbol{A}_{i+1}^T \boldsymbol{\Delta} \boldsymbol{Z}_{i+1} D_{i+1}^{-1} \boldsymbol{Z}_{i+1}^T \boldsymbol{A}_{i+1}, \quad \boldsymbol{\mathcal{Z}}_N = 0. \tag{6.42}$$

  The jth row of $\boldsymbol{\mathcal{Z}}_i$ contains the acceleration energy that the jth unit constraint force has generated (up to now in the recursion) against the accelerations generated by all constraint forces. $\boldsymbol{A}_{i+1}$ is what is felt of the unit constraint *forces* behind the current joint; the multiplication with $\boldsymbol{Z}_{i+1} D_{i+1}^{-1} \boldsymbol{Z}_{i+1}^T$ results in the *acceleration* generated at this joint by those constraint forces; and the multiplication with $\boldsymbol{A}_{i+1}^T$ yields the acceleration energy contributions at this joint. The minus sign comes from the *convention* that the algorithm will compute the *reaction force* to the constraint forces.

- When the recursion arrives at the base ($i = 0$), one can solve for the still unknown constraint force magnitudes $\boldsymbol{\nu}$, via the final constraint acceleration energy balance:

$$\boldsymbol{\mathcal{Z}}_0 \, \boldsymbol{\nu} = \boldsymbol{b}_N - \boldsymbol{A}_0^T \ddot{\boldsymbol{X}}_0 - \boldsymbol{\beta}_0. \tag{6.43}$$

  For a robot with a rigidly fixed base, the acceleration $\ddot{\boldsymbol{X}}_0$ consists of gravity only. The numerical complexity of this problem is $\mathcal{O}(m^3)$, with m the number of constraints.

  The matrix $\boldsymbol{\mathcal{Z}}_0$ is symmetric, as is apparent from Eq. (6.42). But, since each recursion in Eq. (6.42) adds a *matrix of rank one* (at least, if the joint over which the recursion runs allows one degree of motion freedom), and starts with a zero matrix, a minimum of m joint degrees of freedom are needed for the invertibility of $\boldsymbol{\mathcal{Z}}_0$, and hence to generate the m constraint forces. Even then, $\boldsymbol{\mathcal{Z}}_0$ can be *singular* (that is, not of full rank, and hence not invertible), which is the mathematical indication that the kinematic chain is physically unable to generate the desired constraint of Eq. (6.36). A *(weighted) pseudo-inverse* solution can provide a set of joint forces that *approximates* the desired acceleration constraints; the weighting takes place in the space of the *constraint magnitudes* $\boldsymbol{\nu}$.

- The outward joint acceleration remains similar to the unconstrained case of Eq. (6.30), except that an extra joint torque $\boldsymbol{A}_i \, \boldsymbol{\nu}$ is added to generate the desired constraint forces:

$$\ddot{q}_i = (d_i + \boldsymbol{Z}_i^T \boldsymbol{M}_{i+1}^a \boldsymbol{Z}_i)^{-1} \left\{ \tau_i - \boldsymbol{Z}_i^T \left(\boldsymbol{F}_{i+1} + \boldsymbol{M}_{i+1}^a \ddot{\boldsymbol{X}}_i + \boldsymbol{A}_{i+1} \, \boldsymbol{\nu} \right) \right\}. \tag{6.44}$$

[1] "*Zwang*" is the German word for acceleration energy [109, 226], and since much of the early publications in the domain of constrained dynamics appeared in the dominant German scientific literature of the early 20th century, the symbol $\boldsymbol{\mathcal{Z}}$ is commonly used in the international literature to denote acceleration energy.

6.7.2 Null space

The *null space* acceleration of the kinematic chain corresponds to the joint forces that generate a zero acceleration at the end-effector, $\ddot{\boldsymbol{X}}_N = 0$. This is a special case of the constrained end-effector in Eq. (6.36):

$$\boldsymbol{A}_N = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \boldsymbol{b}_N = 0. \tag{6.45}$$

Many inverse dynamics applications want to superimpose a null space motion to the outcome of the constrained ID algorithm, for example, in order to add *posture control* for a humanoid robot, i.e., joint forces or accelerations to generate motions of the legs, torso and arms that do not change the desired hand motion constraints, but that serve to keep the whole robot close to a certain posture, for balancing or obstacle avoidance. Such posture control can, in principle, use an *arbitrary* subset of all possible joints, and the algorithm will find the joint forces that can still realise the desired segment constraints; of course, incompatible specifications of posture control and segment constraints can occur.

The presented recursions deal with the above-mentioned null space motions as follows:

- if the null space motion is specified as desired *accelerations* $\ddot{q}_i$ of some of the joints, the outward acceleration recursion will take them into account in a natural way, resulting in the computation of the corresponding joint forces in the later inward recursion step.
- if the null space motion is specified as desired *forces* τ_i at some of the joints, these are added as given forces to the recursion in Eq. (6.41).

Of course, such extra null space joint forces will violate the minimal kinetic energy property of the presented algorithm, but that violation is then *desired* by the robot task, and not a "bug" of the algorithm.

The presented algorithm does not provide any help in choosing what joint accelerations or forces are appropriate for a desired posture control.

6.7.3 Multiple constraints in a tree

The recursions of the previous Section apply to *serial* kinematic chains, and with a constraint on only one segment. Extending this to multiple constraints on different segments in a tree-structured kinematic chain does not change the recursions in Eq. (6.40) and Eq. (6.41) of, respectively, the constraint force matrix $\boldsymbol{A}$ and the accumulated acceleration energy β of each individual constraint, but involves some extensions for the acceleration energy coupling matrix $\boldsymbol{\mathcal{Z}}$ and for the computation of the constraint force magnitudes $\boldsymbol{\nu}$. The following paragraphs explain the extensions for the case of only two fused constraints, but the procedure holds for any number of constraints:

- When a constraint applies on a segment that is not a leaf segment, the recursions can start from that segment, because more distal segments will not contribute to the constraint satisfaction or violation.
- When the recursion of an k-dimensional constraint c reaches a segment where it must be joined with the recursion of an l-dimensional constraint d, both acceleration constraint coupling matrices $\boldsymbol{\mathcal{Z}}^c$ and $\boldsymbol{\mathcal{Z}}^d$, Eq. (6.42), are fused into a $(k+l) \times (k+l)$-dimensional constraint matrix $\boldsymbol{\mathcal{Z}}^{cd}$:

$$\boldsymbol{\mathcal{Z}}^{cd} = \begin{pmatrix} \boldsymbol{\mathcal{Z}}^c & 0 \\ 0 & \boldsymbol{\mathcal{Z}}^d \end{pmatrix}, \tag{6.46}$$

  and the recursion of $\boldsymbol{\mathcal{Z}}^{cd}$ continues further as in Eq. (6.42), but now with the $6 \times (k+l)$ constraint force matrix $\boldsymbol{A}^{cd}$:

$$\boldsymbol{A}^{cd} = \begin{pmatrix} \boldsymbol{A}^c & \boldsymbol{A}^d \end{pmatrix}. \tag{6.47}$$

  The physical interpretation of $\boldsymbol{\mathcal{Z}}^{cd}$ is clear: its jth row contains the acceleration energy that the jth unit constraint force has generated (up to now in the recursion) against the accelerations generated by all $k+l$ constraint forces. Indeed, the extra joint forces needed to realize the constraint forces in c also influence the constraint forces in d, and vice versa.

- At the base, the extended version of Eq.(6.43) must be solved:

$$\boldsymbol{\mathcal{Z}}_0^{cd} \begin{pmatrix} \boldsymbol{\nu}^c \\ \boldsymbol{\nu}^d \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_N^c - \boldsymbol{\beta}_0^c \\ \boldsymbol{b}_N^d - \boldsymbol{\beta}_0^d \end{pmatrix} - (\boldsymbol{A}_0^{cd})^T \ddot{\boldsymbol{X}}_0. \tag{6.48}$$

  The constraints can be satisfied if $\boldsymbol{\mathcal{Z}}_0^{cd}$ is of full rank, and can be inverted. The numerical complexity of this problem is $\mathcal{O}(m^3)$, with m the *total* number of constraints ($m = k + l$ in the case above).

6.7.4 Free-floating base

If the base of the kinematic chain is not attached to the ground, but freely floating, such as is the case with a robot in space, the base acceleration $\ddot{\boldsymbol{X}}_0$ is not imposed, but is a *result* of the natural dynamics of the robot. It follows straightforwardly from Newton's law:

$$\ddot{\boldsymbol{X}}_0 = -\left(\boldsymbol{M}_0^a\right)^{-1} \left(\boldsymbol{F}_0 + \boldsymbol{A}_0 \boldsymbol{\nu}\right). \tag{6.49}$$

In other words, it moves under the combined action of (i) the physical forces, and (ii) the (virtual) "constraint forces".
(TODO: explain how the *conservation of momentum* is taken into account, [172].)

6.7.5 General base constraints

In this case, the base is subjected to a set of acceleration constraints:

$$\boldsymbol{A}_0^T \ddot{\boldsymbol{X}}_0 = \boldsymbol{b}_0. \tag{6.50}$$

$\boldsymbol{A}_0$ is a $k \times 6$ matrix, with each column interpreted as a constraint force on the base. The base acceleration $\ddot{\boldsymbol{X}}_0$ is now the result of an optimization problem: (TODO)

6.7.6 Local kinematic loops

(TODO) Definition of a *local "loop closure" constraint* (= involves only constraints that can be solved by an m-dimensional set of coupled equations (if there are m joints involved in the local loop), where m is "much smaller" than the number n of bodies in the chain. It's impossible to put a limit on when the ratio m/n remains "small".
[219]

6.7.7 Inequality constraints

(TODO)

6.8 Acceleration constraints—Priority-based approach

This Section computes the inverse dynamics for a kinematic chain with specified accelerations at each of the end-effectors, and at an arbitrary number of other segments.
[130, 131, 133, 132]
(TODO: add full details)

6.8.1 Inequality constraints

(TODO)

6.9 Euler-Lagrange dynamics

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 segments in the chain, and the Cartesian forces exerted on all segments. This involves a non-minimal number of variables, since each segment has only *one* degree of freedom with respect to its neighbours, while the Cartesian velocities and forces for each segment 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*, [140]. 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*

$$I = \int_{t_1}^{t_2} \mathcal{L}\, dt \tag{6.51}$$

reach an *extremal* value (i.e., a local minimum or maximum), [97, 98, 158, 223, 230, 272, 289].

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:

$$\mathcal{L} = T - V. \tag{6.52}$$

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:

$$\mathcal{L} = T - V + W. \tag{6.53}$$

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), [60]) in optics, and Maupertuis' *Principle of Least Action* (Pierre Louis Moreau de Maupertuis (1698–1759), [63, 64]). The precise contents of the word "action" changed over time (Lagrange, for example, used the product of distance and momentum as the "action", [140]) 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 segments 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 segments, 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 (6.51) 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 [236], 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, [73]. 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 transformed principle is somewhat less general than Hamilton's principle (Euler's PDEs are *necessary*, but not sufficient conditions!) but 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.

6.9.1 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., [54, 79, 84, 230]. (Some textbooks derive the opposite direction, i.e., they deduce Hamilton's principle from Lagrange's equations. This proves that both are fully equivalent.) Assume that the extremum value of the action integral is given by

$$I = \int_{t_1}^{t_2} \mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t)\, dt, \tag{6.54}$$

with $\mathcal{L} = T - V$ the desired Lagrangian function we are looking for. $\mathcal{L}$ is a function of the n generalised coordinates $\boldsymbol{q} = (q_1, \ldots, q_n)$, 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:

$$\Phi(\boldsymbol{\epsilon}) = \int_{t_1}^{t_2} \mathcal{L}(\boldsymbol{q} + \boldsymbol{\epsilon}^T \boldsymbol{r}, \dot{\boldsymbol{q}} + \boldsymbol{\epsilon}^T \dot{\boldsymbol{r}}, t)\, dt, \tag{6.55}$$

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:

$$0 = \epsilon_1 \left(\frac{\partial \Phi}{\partial \epsilon_1}\right)_{\epsilon_1 = 0} + \epsilon_2 \left(\frac{\partial \Phi}{\partial \epsilon_2}\right)_{\epsilon_2 = 0} + \cdots + \epsilon_n \left(\frac{\partial \Phi}{\partial \epsilon_n}\right)_{\epsilon_n = 0} \tag{6.56}$$

$$= \int_{t_1}^{t_2} \left(\epsilon_1 \left(\frac{\partial \mathcal{L}}{\partial q_1} r_1 + \frac{\partial \mathcal{L}}{\partial \dot{q}_1} \dot{r}_1 \right) + \epsilon_2 \left(\frac{\partial \mathcal{L}}{\partial q_2} r_2 + \frac{\partial \mathcal{L}}{\partial \dot{q}_2} \dot{r}_2 \right) + \cdots + \epsilon_n \left(\frac{\partial \mathcal{L}}{\partial q_n} r_n + \frac{\partial \mathcal{L}}{\partial \dot{q}_n} \dot{r}_n \right) \right) dt. \tag{6.57}$$

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

$$\epsilon_i \frac{\partial \mathcal{L}}{\partial \dot{q}_i} r_i \,|_{t_1}^{t_2} + \epsilon_i \int_{t_1}^{t_2} r_i \left(\frac{\partial \mathcal{L}}{\partial q_i} - \frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}_i} \right) dt. \tag{6.58}$$

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):

$$\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = 0, \quad i = 1, \ldots, n, \qquad \text{or, in vector form,} \quad \frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{q}} = \boldsymbol{0}. \tag{6.59}$$

6.9.2 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. 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, the envisaged application, or tradition and previous work in a certain domain. However, some general remarks can be made:

- *Recursive* Euler-Lagrange algorithms have been developed, [236], resulting in exactly the same practically useful equations as those that were derived earlier in the literature via the Newton-Euler approach (and which later Sections in this text will describe), such that the historical objection against using the Euler-Lagrange approach because of efficiency reasons has lost much its initial motivation.

- Hamilton's Principle is (by construction) independent of the mathematical representation used, hence the Euler-Lagrange equations derived from it are *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 segments.

- 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, [289].

- The Newton-Euler method uses *vector quantities* (Cartesian velocities and forces), while the Euler-Lagrange method works with *scalar quantities* (energies).

- For a non-redundant six degrees of freedom serial manipulator, the six joint positions $\boldsymbol{q}$ could be replaced, as a choice of generalized coordinates, by six Cartesian position parameters $\boldsymbol{x}$ of the end-effector. However, these generalised coordinates $\boldsymbol{x}$ are only a *local* representation, hence the corresponding Euler-Lagrange approach is not globally valid (as is the Newton-Euler approach).

6.9.3 Constrained systems

For mobile robots, the validity of d'Alembert's and Gauss' principles is easily understood: the constraint forces work along the direction of the wheel axle, while the robot does not move along this direction; hence the work performed by these constraint forces is zero. The constraint forces in Eq. (4.92) can be straightforwardly included in the Euler-Lagrange equations of the system:

$$\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{q}} = \boldsymbol{\tau} + \boldsymbol{A}^T(\boldsymbol{q}) \boldsymbol{\phi}. \tag{6.60}$$

The ϕ_i are the *Lagrange multipliers* of the constraints.

6.10 Numerical solvers

TODO: *Differential Algebraic Equations* (DAE), *ordinary differential equations* (ODE), *partial differential equations* (PDE), index of DAE, Baumgarte stabilization; internal constraints, end-point constraints. Higher order constraint specifications assume the lower order constraints are already satisfied. [182]

Many numerical techniques assume "stiff" constraints, but that is not always a good approximation; and helping the "blind" solvers with physically relevant information about what is soft or hard can help tremendously; especially when the calculations have to be performed in a feedback control loop where some sort of positioning errors have to be allowed anyway.

Chapter 7

Serial and tree-structured chains

This Chapter looks at the *specific properties* (i.e., more specific than the generic properties presented in the previous Chapter, such as closed-form algorithms) for the *displacement*, *velocity*, *acceleration* and *force* characteristics of *serial* and *tree*-structured kinematics chains. Both families of kinematic chains have most of their generic physical and computational properties in common, since they have no *closed loops*.

7.1 Serial robot designs

For serial chains, the mapping from joint positions to end-effector pose is "easy," while the inverse mapping is "difficult." At least, in the case that the chain has an *arbitrary* kinematic structure; i.e., the relative position and orientation of subsequent joint axes can be anything. To avoid such computational complexities in the inverse kinematics is the reason why most industrial robots have specially engineered designs, often at the cost of increased *mechanical* complexity. The mathematically simplest designs (of practical interest) make use of a *spherical wrist* (such as the one depicted in Fig. 7.1). Most robots use 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, [201], Donald L. Pieper (1941–) presented a very practically relevant result:

> The inverse kinematics of any serial manipulator with *six revolute* joints, and with *three consecutive joints intersecting* (Fig. 7.1), can be solved in *closed-form*, i.e., *analytically*.

This result does not (necessarily) hold for robots with more than six revolute joints, such as the *Light-Weight Arm* from KUKA (Fig. 7.5), the *Whole Arm Manipulator* from Barrett Technologies (Fig. 7.6), or the *PR2* from Willow Garage (Fig. 7.7).

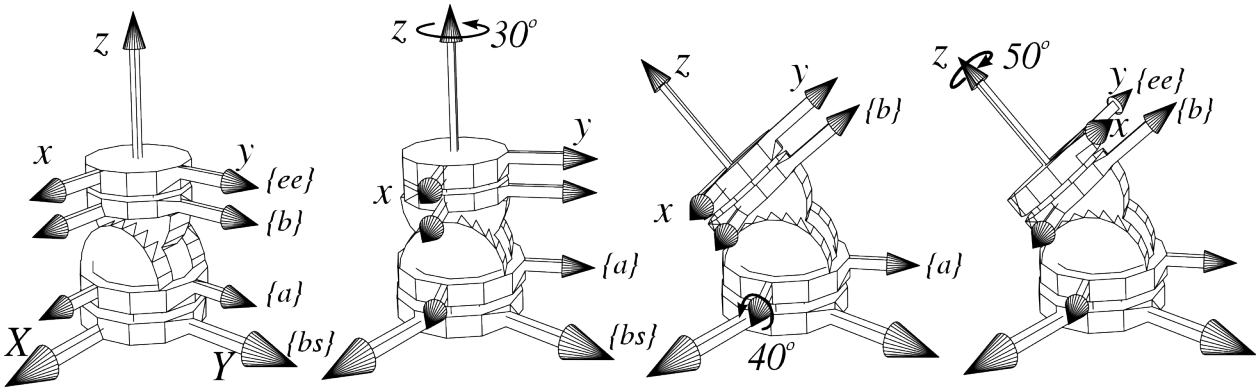

Figure 7.1: The three motion capabilities of a spherical ZXZ wrist. This terminology comes from the observation that the depicted wrist makes rotations about the Z, X and Z axes of the subsequent joints.

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. 7.2), all industrial manipulators had at least one prismatic joint [281] (see e.g., [271] for an impressively large catalogue)

while since then, most industrial robots are *wrist-partitioned 6R* manipulators, such as shown in Figures 7.3 and 7.4. These *6R* robots have six revolute joints, and their last three joint axes intersect orthogonally, i.e., they form a *spherical wrist* such as, for example, the *ZXZ* wrist whose motion capabilities are illustrated in Fig. 3.7. Hence, they can achieve any possible orientation.

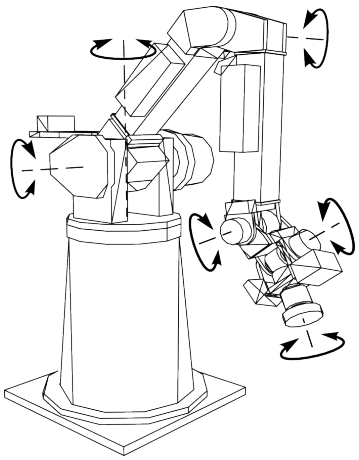

Figure 7.2: The *Cincinnati Milacron* T^3 serial robot.

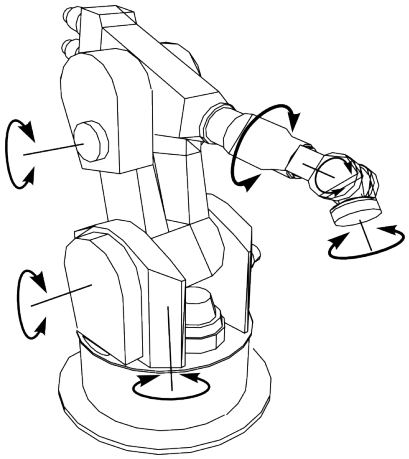

Figure 7.3: A *Kuka-160* serial robot.

As Pieper proved, this construction leads to a *decoupling* of the position and orientation kinematics, for the forward as well as the inverse problems. The inverse solution for the three wrist joints is a copy of the inverse Euler angle problem of Sect. 3.13; the remaining three joints are then found by solving a polynomial of, at most, fourth order, whatever their kinematic structure is. The extra structural simplifications (i.e., parallel or orthogonal axes) introduced in the serial robots of, for example, Figures 7.3 and 7.4, lead to even simpler solutions (Sect. 7.10). The simplest kinematics are found in the *SCARA* (*Selectively Compliant Assembly Robot Arm*) design, Fig. 7.8. This design has three vertical revolute joint axes, and one vertical prismatic joint at the end. SCARA 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.

Hybrid designs. A last industrially important class of "serial" robot arms are the *gantry* robots, Fig. 7.9. 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 *XYZ* construction is made parallel, to make it very stiff and hence accurate. (That's why this design is often seen in

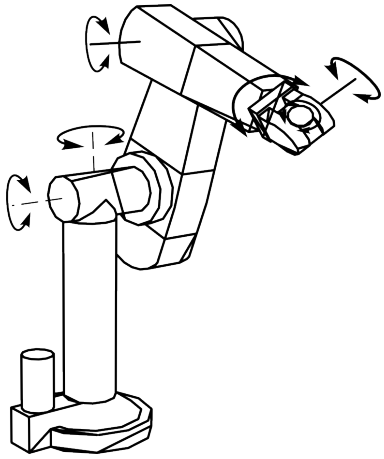

Figure 7.4: A *Staubli* (formerly *Unimation*) "PUMA" serial robot.

Figure 7.5: A *KUKA* "Light-Weight Robot" with seven degrees of freedom, as an example of the *313* serial structure.

metal cutting machines.) But geometrically speaking, the gantry construction has the same simple form as that of three *serial* prismatic joints, which is why it is discussed in this Chapter. In large industrial applications (such as welding of ship hulls or other large objects) a serial manipulator is often attached to a two or three degrees of freedom gantry structure, in order to combine the workspace and dexterity advantages of both kinematic structures.

Design characteristics. The examples above illustrate the common design characteristics of (most) industrial 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. Serial robots 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 segments must be stiff: flexible segments cause deformations, and hence position and orientation errors at the end-point.

4. Simplicity of the forward and inverse position and velocity kinematics has always been one of the major design criteria for industrial manipulator arms. Hence, almost all of them have a very special kinematic

Figure 7.6: A *Whole Arm Manipulator* from *Barrett Technologies*, with seven degrees of freedom, as another example of the *313* serial structure.

Figure 7.7: The *PR2* mobile manipulator of Willow Garage has two arms with seven degrees of freedom, but not of the *313* structure.

structure, with a majority having the *321 design* of Fig. 7.11. The special kinematic structures have efficient closed-form solutions (discussed in later Sections) because they allow for the *decoupling* of the position and orientation kinematics. The geometric feature that generates this decoupling is the *intersection* of joint axes.

5. Most robots have gearboxes between the joints and the actuators that drive these joints. Typical gear ratios for industrial robots are in the range 1/10–1/500, with the motors making more revolutions than the joints. Hence, (i) the position of the *joint* is different from the position of the *motor*, and (ii) the *inertia* of the rotor in the joint actuator is sometimes more important than the inertia of the segments interconnected by the joint.

7.2 321 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, most industrial robot arms now have a kinematic structure as shown in Fig. 7.11. (Vic Scheinman of Stanford University was, to the best of the authors' knowledge, the first to come up with this design, around 1970, 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 structure can be given a segment frame transformation convention that is much simpler than the Denavit-Hartenberg or Hayati-Roberts conventions, [75]: its geometry is determined by orthogonal and parallel joint axes, and by only four segment lengths l_1, l_2, l_3 and l_6, because the wrist segment lengths l_4 and l_5 are zero. In this simpler convention, the reference frames are chosen to be all parallel when the robot is in its fully upright configuration. This configuration is defined to be the *kinematic zero* position in the rest of this text, 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 different 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 7.4), have a kinematic structure that deviates a little bit from the 321 structure of Figure 7.11, [56, 240, 281]:

1. *Shoulder offset*: frame {3} in Figure 7.11 is shifted a bit along the X-axis. This brings the elbow off-centre with respect to the line of joint 1 (Fig. 7.12).

2. *Elbow offset*: frame {4} in Figure 7.11 is shifted a bit along the Y-axis. This brings the wrist centre point off-centre with respect to the forearm (Fig. 7.12).

The reasons for the offsets will become clear in the Section on singularities (Sect. 7.19): the offsets mentioned above move the singular positions of the robot away from places in the workspace where they are likely to cause problems. Since the late 1990s, a third type of offset has been commonly used:

3. *Base offset*: frame {2} in Figure 7.11 is shifted a bit forward along the Y-axis (Fig. 7.13). This allows to make the workspace of the robot a bit larger by only increasing the load on the first axis, i.e., without

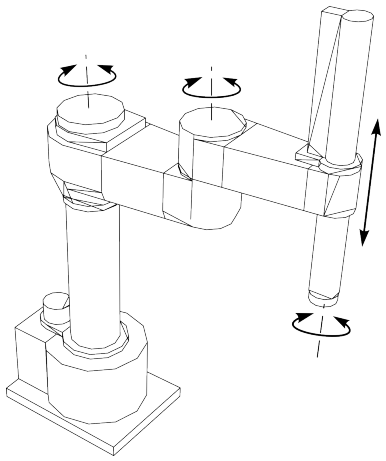

Figure 7.8: An *Adept SCARA* robot.

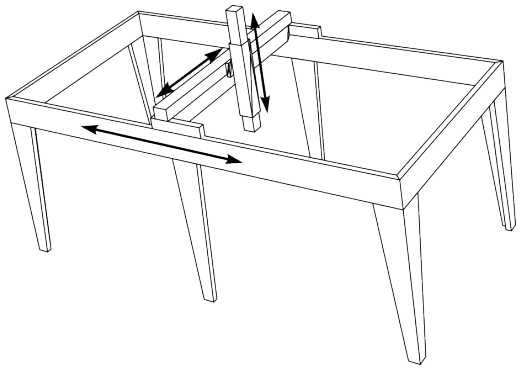

Figure 7.9: A gantry robot. (Only the first three prismatic degrees of freedom are shown.)

having to extend the forearm of the robot which would increase the load on the second and third joints too. A base offset has no influence on the singularity properties of the kinematic design.

7.3 313 kinematic structure

More and more modern robot arms get *seven* revolute joints, e.g. Figs 7.5–7.7. One of the popular designs is a small extension of the 321 structure of Sect. 7.2, as depicted in the figure of the KUKA Light-Weight Arm, Fig 7.5:

7.4 General FPK: segment transform algorithm

The easiest approach to calculate the Forward Position Kinematics (FPK) is to apply the recursive composition formula, Eq. (5.9), from the base frame $\{bs\}$ to the end-effector frame $\{ee\}$:

$$ {}^{ee}_{bs}\boldsymbol{T} = {}^{0}_{bs}\boldsymbol{T}\ {}^{1}_{0}\boldsymbol{T}(q_1)\ {}^{2}_{1}\boldsymbol{T}(q_2)\dots\ {}^{n}_{n-1}\boldsymbol{T}(q_n)\ {}^{ee}_{n}\boldsymbol{T}. \tag{7.1} $$

Each segment transform can be constructed easily for any chosen segment frame definition; for example, from the Denavit-Hartenberg definition (although that definition has no particular advantages at all in this case). This equation can be replaced by a sequence of matrix exponentiations, in which each matrix represents a linear or angular velocity generated by one of the prismatic or revolute joints in the serial arm. Equation (7.1) is then called the *product of exponentials* formula for a serial robot, [28, 173, 192].

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

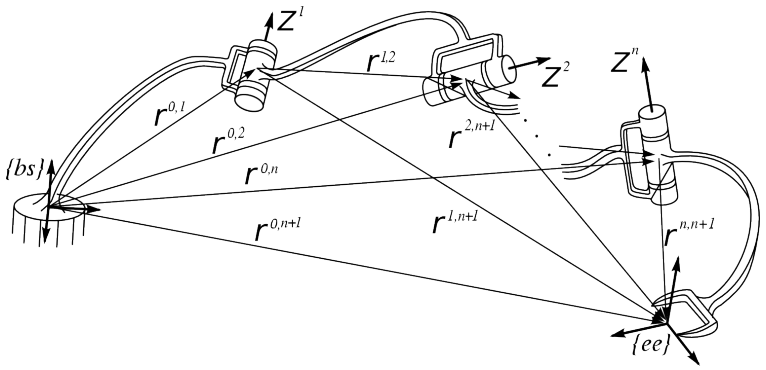

Figure 7.10: Notations used in the geometrical model of a serial kinematic chain (with only one degree-of-freedom joints). The model represents: reference frames, and positions and orientations of joint angles.

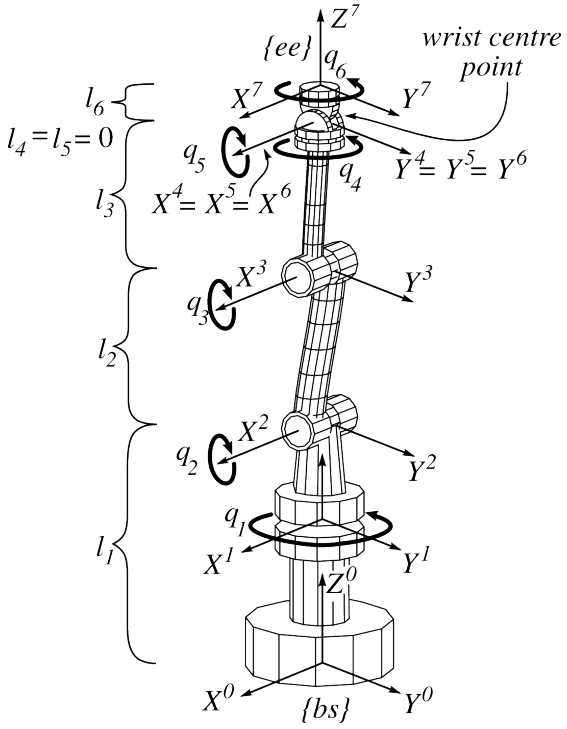

Figure 7.11: 321 kinematic structure in the "zero" position: all segment frames are parallel and all origins lie on the same line.

in a computer program, one should, of course, not code the complete matrix multiplications of Eq. (7.1), since (i) the last rows of the homogeneous transformation matrices are zeros and ones, and (ii) many robots have a kinematic structure that generates many more zeros in the rest of the matrices too.

7.5 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 treat closed-form 321 kinematics) explains the very efficient "closed-form" algorithms that have been developed for this serial design, starting by "splitting" the manipulator at the *wrist centre point* (reference frame {4} in Fig. 7.11). This has the following advantages:

1. The position and linear velocity of the wrist centre point are completely determined by the first three joint

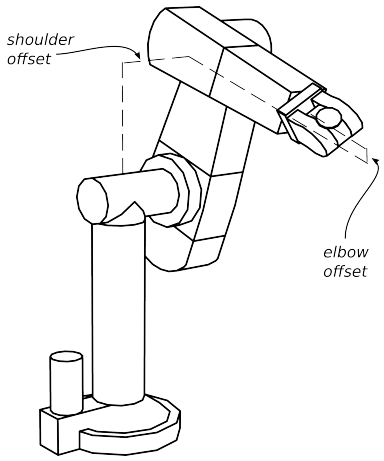

Figure 7.12: A 321 serial manipulator with *shoulder* and *elbow* offsets.

Figure 7.13: KUKA KR150: a 321 serial manipulator with a *base offset*.

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 easy to find from the three wrist joint angles.

3. The relative pose of the end-effector frame $\{7\}$ with respect to the last wrist frame $\{6\}$ is a *constant* translation along the Z^6-axis. A similar relationship holds between the frames on the base and on the first segment.

The following procedure applies this approach to the forward position kinematics:

Closed-form FPK

Step 1 The closed-form forward *orientation* kinematics of the 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 $\{6\}$, 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, is:

$$ {}^6_4\boldsymbol{T} = \begin{pmatrix} {}^6_4\boldsymbol{R} & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} = \begin{pmatrix} c_6c_4 - s_6c_5s_4 & -s_6c_4 - c_6c_5s_4 & -s_5s_4 & 0 \\ c_6s_4 + s_6c_5c_4 & -s_6s_4 + c_6c_5c_4 & s_5c_4 & 0 \\ -s_6s_5 & -c_6s_5 & c_5 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{7.2} $$

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. 7.14):

$$ d^v = c_2l_2 + c_{23}l_3, \qquad d^h = s_2l_2 + s_{23}l_3, \tag{7.3} $$

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 $q_2 + q_3$:

$$ {}^4_0\boldsymbol{R} = \begin{pmatrix} c_1 & -s_1 & 0 \\ s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{23} & -s_{23} \\ 0 & s_{23} & c_{23} \end{pmatrix} = \begin{pmatrix} c_1 & -s_1c_{23} & s_1s_{23} \\ s_1 & c_1c_{23} & -c_1s_{23} \\ 0 & s_{23} & c_{23} \end{pmatrix}. \tag{7.4} $$

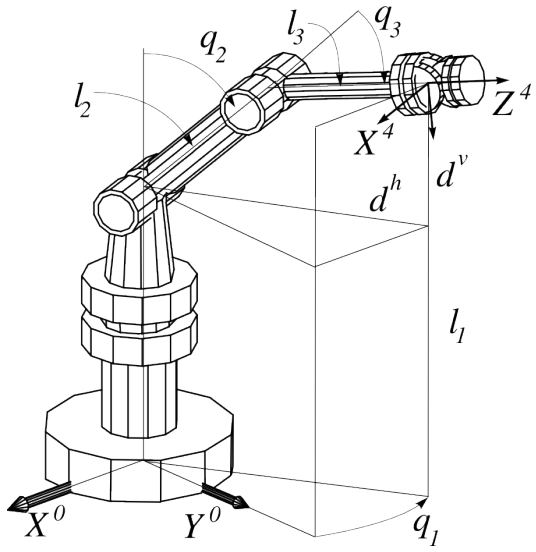

Figure 7.14: Kinematics of first three joints of the 321 manipulator (Fig. 7.11).

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:

$$ {}^{7}_{6}\boldsymbol{T} = \begin{pmatrix} {}^{7}_{6}\boldsymbol{R} & l_6\boldsymbol{e}_z \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_6 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{7.5} $$

Step 4 Hence, the total orientation ${}^{ee}_{bs}\boldsymbol{R}$ follows from Eqs. (7.2), (7.4), and (7.5):

$$ {}^{ee}_{bs}\boldsymbol{R} = {}^{7}_{0}\boldsymbol{R} = {}^{4}_{0}\boldsymbol{R}\ {}^{6}_{4}\boldsymbol{R}\ {}^{7}_{6}\boldsymbol{R}. \tag{7.6} $$

Step 5 The position of the wrist centre (i.e., the origin of $\{4\}$ with respect to the base $\{0\}$) is

$$ {}_{bs}\boldsymbol{p}^{wr} = {}_{0}\boldsymbol{p}^{wr} = \begin{pmatrix} c_1 d^h \\ s_1 d^h \\ l_1 + d^v \end{pmatrix}, \tag{7.7} $$

and the position of the end-effector (i.e., the origin of $\{ee\}$ with respect to the base $\{0\}$) is

$$ {}_{bs}\boldsymbol{p}^{ee} = {}_{bs}\boldsymbol{p}^{wr} + {}^{ee}_{bs}\boldsymbol{R}\ (0\,0\,l_6)^T. \tag{7.8} $$

Step 6 Equations (7.6) and (7.8) yield the final result:

$$ {}^{ee}_{bs}\boldsymbol{T} = \begin{pmatrix} {}^{ee}bs\boldsymbol{R} & {}_{bs}\boldsymbol{p}^{ee} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix}. \tag{7.9} $$

7.6 General Forward velocity kinematics: velocity recursion

The forward velocity kinematics (FVK) solves the following problem:

> *Given the vectors of (i) joint positions* $\boldsymbol{q} = (q_1\ \ldots\ q_n)^T$ *and (ii) joint velocities* $\dot{\boldsymbol{q}} = (\dot{q}_1\ \ldots\ \dot{q}_n)^T$, *and the model of the kinematic chain, 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. This Section explains how to calculate the Jacobian matrix (which is the linear mapping from joint velocities to end-effector velocity) of a serial chain, starting from the knowledge about the

current joint positions. The following procedure works for any serial structure with an arbitrary number of n joints [190]. 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:

$$i = 0, \quad \text{and} \quad {}_0\mathbf{t}^0 = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}. \tag{7.10}$$

Step 1 Recursion $i \to i+1$, until $i = n$:

Step 1.1 Transformation of the twist ${}_i\mathbf{t}^i$ to the next joint:

$${}_{i+1}\mathbf{t}^i = {}^i_{i+1}\boldsymbol{S}\ {}_i\mathbf{t}^i, \tag{7.11}$$

where the screw transformation matrix ${}^i_{i+1}\boldsymbol{S}$ is constructed from the (known) segment transform ${}^i_{i+1}\boldsymbol{T}$.

Step 1.2 Addition of the contribution of joint $i+1$:

$${}_{i+1}\mathbf{t}^{i+1} = {}_{i+1}\mathbf{t}^i \ + \ {}_{i+1}\boldsymbol{J}_{i+1}\ \dot{q}_{i+1}. \tag{7.12}$$

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:

$${}_w\mathbf{t}^{ee} = {}^{ee}_w\boldsymbol{S}\ {}_{ee}\mathbf{t}^{ee}. \tag{7.13}$$

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:

$${}_w\boldsymbol{J}_i = {}^w_1\boldsymbol{S}\ {}^1_0\boldsymbol{S}\ {}^2_1\boldsymbol{S}\ \cdots\ {}^i_{i-1}\boldsymbol{S}\ {}_i\boldsymbol{J}_i.$$

Variations on this FVK algorithm have appeared in the literature, differing only in implementation details to make the execution of the algorithm more efficient.

7.7 Closed-form FVK for 321 structure

For the 321 kinematic structure, more efficient closed-form solutions exist, [75, 152, 212, 213, 267] and [120]. 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, [68].)

Closed-form FVK

Step 1 The wrist is of the ZXZ type, so 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. (7.2). In total, this yields

$${}_4\boldsymbol{J}_{456} = \begin{pmatrix} {}_4\boldsymbol{J}_4 & {}_4\boldsymbol{J}_5 & {}_4\boldsymbol{J}_6 \end{pmatrix} = \begin{pmatrix} 0 & c_4 & -s_5 s_4 \\ 0 & s_4 & s_5 c_4 \\ 1 & 0 & c_5 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{7.14}$$

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 7.14 yields

$$ {}_4\boldsymbol{J}_{123} = \begin{pmatrix} {}_4\boldsymbol{J}_1 & {}_4\boldsymbol{J}_2 & {}_4\boldsymbol{J}_3 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ -s_{23} & 0 & 0 \\ c_{23} & 0 & 0 \\ -d^h & 0 & 0 \\ 0 & l_2c_3 + l_3 & l_3 \\ 0 & l_2s_3 & 0 \end{pmatrix}. \tag{7.15} $$

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 ${}^4_{bs}\boldsymbol{S}$:

$$ {}_{bs}\boldsymbol{J} = {}^4_{bs}\boldsymbol{S}\ {}_4\boldsymbol{J}. \tag{7.16} $$

${}^4_{bs}\boldsymbol{S}$ is straightforwardly derived from the solution to the forward position kinematics of the robot (Sect. 7.5).

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:

$$ \begin{aligned} \det({}_4\boldsymbol{J}) &= \det \begin{pmatrix} -d^h & 0 & 0 \\ 0 & l_2c_3 + l_3 & l_3 \\ 0 & l_2s_3 & 0 \end{pmatrix} \det \begin{pmatrix} 0 & c_4 & -s_5s_4 \\ 0 & s_4 & s_5c_4 \\ 1 & 0 & c_5 \end{pmatrix} \\ &= -d^h l_2 l_3 s_3 s_5. \end{aligned} \tag{7.17} $$

Note that the determinant of the Jacobian is independent of the reference frame with respect to which it is calculated:

$$ {}_f\boldsymbol{J} = {}^i_f\boldsymbol{S}\ {}_i\boldsymbol{J} \Rightarrow \det({}_f\boldsymbol{J}) = \det({}^i_f\boldsymbol{S})\ \det({}_i\boldsymbol{J}), \tag{7.18} $$

and $\det(\boldsymbol{S}) = \det^2(\boldsymbol{R}) = 1$.

7.8 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 for serial chains: the same end-effector pose can be reached in several *configurations*, corresponding to distinct joint position vectors. It can be proven, [142, 209], that a 6R manipulator (a serial chain with six revolute joints, as in Figs 7.3, 7.4, and 7.2), with a *completely general* geometric structure, has *sixteen* different inverse kinematics solutions, 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 [75]. Some older references describe similar solution approaches but in less detail, [119, 201, 221].

The IK of a serial arm is 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. However, 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 General IPK: Newton-Raphson iteration

Inverse position kinematics for serial robot arms with a *completely general* kinematic structure are solved by iterative procedures, based on the Newton-Raphson approach, [201, 247]. However, this simple approach works only for chains with *six* joints, and away from singularities, Sect. 5.8.2; the other cases (that is, with more or less than six joints and/or near singularities), are dealt with in Sect. 5.10.

Numerical IPK

Step 1 Start with an *estimate* $\hat{\boldsymbol{q}} = (\hat{q}_1 \dots \hat{q}_6)^T$ 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:

$$\boldsymbol{T}(\boldsymbol{q}) = \boldsymbol{T}(\hat{\boldsymbol{q}})\ \boldsymbol{T}_\Delta(\Delta\boldsymbol{q}). \tag{7.19}$$

$\Delta\boldsymbol{q} = \boldsymbol{q} - \hat{\boldsymbol{q}}$ is the joint position increment to be solved by the iteration. Solving for $\boldsymbol{T}_\Delta(\Delta\boldsymbol{q})$ yields

$$\boldsymbol{T}_\Delta(\Delta\boldsymbol{q}) = \boldsymbol{T}^{-1}(\hat{\boldsymbol{q}})\boldsymbol{T}(\boldsymbol{q}). \tag{7.20}$$

Step 3 The coordinate expression of the infinitesimal pose $\boldsymbol{T}_\Delta(\Delta\boldsymbol{q})$ is:

$$\boldsymbol{T}_\Delta = \begin{pmatrix} 1 & -\delta_z & \delta_y & d_x \\ \delta_z & 1 & -\delta_x & d_y \\ -\delta_y & \delta_x & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{7.21}$$

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. (7.21). On the other hand, it depends linearly on the joint increment $\Delta\boldsymbol{q}$ through the Jacobian matrix $\boldsymbol{J}(\hat{\boldsymbol{q}})$, Eq. (5.23):

$$\mathbf{t}_\Delta(\hat{\boldsymbol{q}}) = \boldsymbol{J}(\hat{\boldsymbol{q}})\Delta\boldsymbol{q} + \mathcal{O}(\Delta\boldsymbol{q}^2). \tag{7.22}$$

$\boldsymbol{J}(\hat{\boldsymbol{q}})$ is calculated by the numerical FVK algorithm in Sect. 7.6.

Step 4 Hence, the joint increment $\Delta\boldsymbol{q}$ is approximated by

$$\Delta\boldsymbol{q} = \boldsymbol{J}^{-1}(\hat{\boldsymbol{q}})\,\mathbf{t}_\Delta(\hat{\boldsymbol{q}}). \tag{7.23}$$

The inverse of the Jacobian matrix exists only when the robot arm has *six* independent joints. Section 5.9 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. (7.23). Numerically more efficient and stable algorithms exist, [85, 247], 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*, [210, 255, 269].

7.10 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, [75]:

Closed-form IPK

Step 1 The position of the wrist centre point is simply given by the inverse of Eq. (7.8):

$$ {}_{bs}\boldsymbol{p}^{wr} = {}_{bs}\boldsymbol{p}^{ee} - {}^{ee}_{bs}\boldsymbol{R}\,(0\,0\,l_6)^T. \tag{7.24}$$

Step 2 Hence, the first joint angle is

$$ q_1 = \text{atan2}({}_{bs}\boldsymbol{p}^{wr}_x, \pm {}_{bs}\boldsymbol{p}^{wr}_y). \tag{7.25}$$

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*."

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. 7.14:

$$ d^h = \sqrt{({}_{bs}\boldsymbol{p}^{wr}_x)^2 + ({}_{bs}\boldsymbol{p}^{wr}_y)^2}, \quad d^v = {}_{bs}\boldsymbol{p}^{wr}_z - l_1. \tag{7.26}$$

Step 4 Now, look at the planar triangles formed by the second and third segments (Fig. 7.14).

Step 4.1 The cosine rule gives

$$ q_3 = \pm \arccos\left(\frac{(d^h)^2 + (d^v)^2 - (l_2)^2 - (l_3)^2}{2l_2l_3}\right). \tag{7.27}$$

A positive q_3 gives the "*elbow up*" configuration (Fig. 7.15); the configuration with negative q_3 is called "*elbow down*."

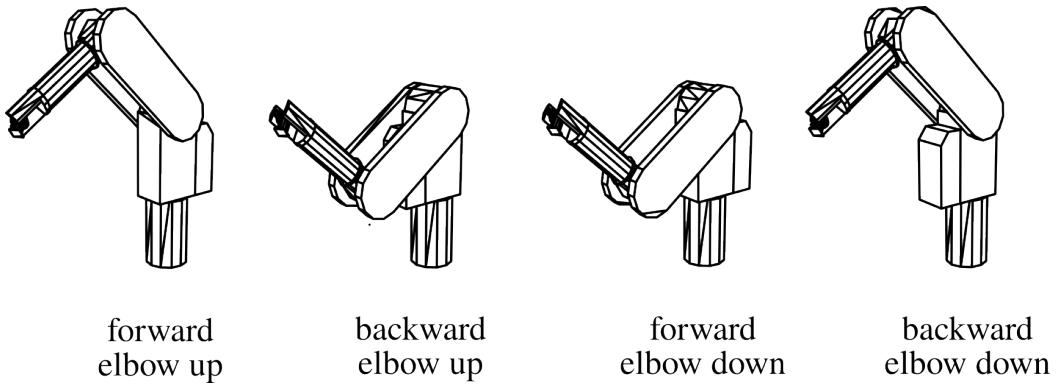

Figure 7.15: Four of the eight configurations corresponding to the same end effector pose, for a 321 type of manipulator. The four other configurations are similar to these four, except for a change in the wrist configuration from "flip" to "no flip." (TODO: make figures more clear, especially for the wrist!)

Step 4.2 The tangent rules yield

$$ q_2 = \text{atan2}\left(d^h, d^v\right) - \alpha, \tag{7.28}$$

with

$$ \alpha = \text{atan2}\left(l_3 s_3, l_2 + l_3 c_3\right). \tag{7.29}$$

Step 5 The inverse position for the *ZXZ* wrist uses ${}^{6}_{4}\boldsymbol{R}$ as input, which is straightforwardly derived from ${}^{7}_{bs}\boldsymbol{R}$ (a known input parameter) and ${}^{4}_{bs}\boldsymbol{R}$ (which can be calculated as soon as the first three joint angles are known):

$$ {}^{6}_{4}\boldsymbol{R} = {}^{7}_{4}\boldsymbol{R} = {}^{bs}_{4}\boldsymbol{R}\, {}^{7}_{bs}\boldsymbol{R}, \tag{7.30} $$

with

$$ {}^{4}_{bs}\boldsymbol{R} = \boldsymbol{R}(Z, q_1)\boldsymbol{R}(X, -q_2 - q_3). \tag{7.31} $$

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. 5.1) corresponds to a particular choice of IPK solution. In total, the 321 manipulator has eight different configurations [263], 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.

7.11 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* $\mathbf{t}^{ee}$*, and the model of the kinematic chain, what is the corresponding vector of joint velocities* $\dot{\boldsymbol{q}} = (\dot{q}_1 \ \ldots \ \dot{q}_n)^T$*?*

An alternative name for the IVK algorithm is the "*resolved rate*" procedure, especially in the context of robot control, [275].

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. The IVK 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. These cases are discussed in Sect. 5.9.

7.12 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:

$$ \dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\boldsymbol{q})\, \mathbf{t}^{ee}. \tag{7.32} $$

As mentioned before, more efficient and robust algorithms calculate $\dot{\boldsymbol{q}}$ without the explicit calculation of the matrix inverse $\boldsymbol{J}^{-1}$, Sect. 5.10.

7.13 Closed-form IVK for 321 structure

The symbolically derived Jacobian for the 321 kinematic structure (Sect. 7.7) turns out to be also easily invertible symbolically, when expressed in the wrist centre frame, [120, 212, 213]:

Step 1 The Jacobian ${}_4\boldsymbol{J}$ (Eqs (7.14) and (7.15)) has a zero 3×3 block in the lower right-hand side:

$$ {}_4\boldsymbol{J} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{0}_3 \end{pmatrix}. \tag{7.33} $$

Step 2 It is then easily checked by straightforward calculation that

$$ {}_4\boldsymbol{J}^{-1} = \begin{pmatrix} \boldsymbol{0}_3 & \boldsymbol{C}^{-1} \\ \boldsymbol{B}^{-1} & -\boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{C}^{-1} \end{pmatrix}. \tag{7.34} $$

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 (7.14)–(7.15):

$$ \det(\boldsymbol{B}) = s_5, \qquad \det(\boldsymbol{C}) = l_2 l_3 d^h s_3. \tag{7.35} $$

Hence,

$$ \boldsymbol{B}^{-1} = \frac{1}{s_5}\begin{pmatrix} s_4c_5 & s_5c_4 & -s_4 \\ -c_5c_4 & s_5s_4 & c_4 \\ s_5 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{C}^{-1} = \begin{pmatrix} -\dfrac{1}{d^h} & 0 & 0 \\ 0 & 0 & \dfrac{1}{l_2 s_3} \\ 0 & -\dfrac{1}{l_3} & -\dfrac{l_2c_3 + l_3}{l_2 l_3 s_3} \end{pmatrix}. \tag{7.36} $$

Step 4 In order to find the joint velocities, one has to *post*-multiply ${}_4\boldsymbol{J}^{-1}$ by ${}^{bs}_{4}\boldsymbol{S}$:

$$ \dot{\boldsymbol{q}} = {}_4\boldsymbol{J}^{-1}\ {}^{bs}_{4}\boldsymbol{S}\ {}_{bs}\mathbf{t} = {}_{bs}\boldsymbol{J}^{-1}\ {}_{bs}\mathbf{t}. \tag{7.37} $$

7.14 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 ("6D force")* $\mathbf{w}^{ee}$ *that acts on the end-effector, and given the model of the kinematic chain, what is the vector of joint forces/torques* $\boldsymbol{\tau} = (\tau_1 \ \ldots \ \tau_n)^T$ *that keeps* $\mathbf{w}^{ee}$ *in static equilibrium?*

This Section presents two equivalent approaches, one via direct projection on the joint axes (Sect. 7.14.1), and one via the principle of conservation of virtual work (Sect. 7.14.2).

7.14.1 Inward force recursion—"Jacobian transpose" projection

While the velocity at each joint *adds* to the overall velocity of the end-effector, a force ("wrench") $\mathbf{w}$ exerted on the end-effector segment 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 ${}^{bs}_{i}\boldsymbol{S}$:

$$ {}_i\mathbf{w} = {}^{bs}_{i}\boldsymbol{S}\ {}_{bs}\mathbf{w}. \tag{7.38} $$

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 ${}^{bs}_{i}\boldsymbol{S}^T$:

$$ {}^{bs}_{i}\boldsymbol{S}_{3\times} = {}^{bs}_{i}\boldsymbol{S}^T_{\times 3} = \begin{pmatrix} {}_{bs}\boldsymbol{e}^i_z \\ \boldsymbol{0} \end{pmatrix}, \quad \text{and} \quad {}^{bs}_{i}\boldsymbol{S}_{6\times} = {}^{bs}_{i}\boldsymbol{S}^T_{\times 6} = \begin{pmatrix} {}_{bs}\boldsymbol{p}^{bs,i} \times {}_{bs}\boldsymbol{e}^i_z \\ {}_{bs}\boldsymbol{e}^i_z \end{pmatrix}, \tag{7.39} $$

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

$$\boldsymbol{\Delta} = \begin{pmatrix} \mathbf{0}_{3\times 3} & \mathbf{1}_{3\times 3} \\ \mathbf{1}_{3\times 3} & \mathbf{0}_{3\times 3} \end{pmatrix} \tag{7.40}$$

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

$$\boldsymbol{\tau} = (\boldsymbol{\Delta}\, \boldsymbol{J})^T \mathbf{w}. \tag{7.41}$$

The $\boldsymbol{\Delta}$ is solely an artifact coming from the choice of mathematical representation, and it disappears if one chooses to represent twists (or wrenches) by six-dimensional coordinate vectors that have their linear and angular components in the other place with respect to the convention followed in this text. So, in the robotics literature you see this relationship most often in the form $\boldsymbol{\tau} = \boldsymbol{J}^T \mathbf{w}$, due to this difference in twist representation: $\boldsymbol{J}^{\text{literature}} = \boldsymbol{\Delta}\, \boldsymbol{J}^{\text{this text}}$.

Equation (7.41) 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.

Mathematically speaking, $\boldsymbol{\Delta J}$ is not a "Jacobian matrix," since wrenches are not the partial derivatives of anything. (They *can* be the *gradient* of a *potential field*, however.)

7.14.2 Conservation of virtual work

The work done by a twist $\mathbf{t}$ of the end-effector against the wrench $\mathbf{w}$ that works on the same end-effector, is the same work as performed by the joint velocities $\dot{\boldsymbol{q}}$ on the joint torques $\boldsymbol{\tau}$: $\mathbf{t}^T \boldsymbol{\Delta} \mathbf{w} = \dot{\boldsymbol{q}}^T \boldsymbol{\tau}$. Together with the Jacobian relationship Eq. (5.23), this relation directly leads to Eq. (7.41).

7.15 Dual twist–wrench bases

For a serial chain with six joints, the Jacobian matrix is a *basis* for the twist spac of the end-effector (Sect. 5.5). A *dual basis* for the wrench space exists 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, whose physical interpretation is as follows: 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, [165]. This text uses the notation $\boldsymbol{G} = (\boldsymbol{G}_1 \ \ldots \ \boldsymbol{G}_6)$ for the matrix of the six dual basis wrenches $\boldsymbol{G}_i$. Its formal definition follows from the following relationship with the Jacobian matrix $\boldsymbol{J}$ of the same robot arm:

$$\boldsymbol{J}\ \boldsymbol{\Delta}\ \boldsymbol{G} = \mathbf{1}_{6\times 6}. \tag{7.42}$$

Since $\boldsymbol{G}$ is a basis for the wrench space, each wrench $\mathbf{w}$ on the end-effector has coordinates $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_6)$:

$$\mathbf{w} = \boldsymbol{G}\ \boldsymbol{\tau}. \tag{7.43}$$

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 segments!). The relation with the "Jacobian transpose" formula for the Inverse Force Kinematics, Eq. (7.41), is also immediately clear:

$$\boldsymbol{G} = (\boldsymbol{\Delta J})^{-T}. \tag{7.44}$$

7.16 Closed-form Forward Force Kinematics for 321 structure

As before, using the wrist centre point of the 321 kinematic structure allows for a solution of the FFK problem (Sect. 5.7.6) by simple inspection, since the partial wrench of each joint is easily found from Fig. 7.14 in Sect. 7.5:

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.

(TODO: figure with the six partial wrenches.)

7.17 Forward acceleration kinematics

(TODO: see [114]...)

7.18 Inverse acceleration kinematics

(TODO: see [114]...)

7.19 Singularities

For serial and tree chains, it's the inverse velocity kinematics that can exhibit singularities (Sect. 5.8), whose physical interpretation is that, at such a *singularity*, the end-effector looses one or more degrees of (Cartesian) *twist* freedom (i.e., instantaneously, the end-effector cannot move in these directions). Mathematically speaking, the Jacobian matrix $\boldsymbol{J}$ *looses* rank. 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. These singularities are *physical*, in the sense that they have a physical origin, so they are found in any (physically faithful) mathematical representation used to describe the kinematics. However, in addition to the physical singularities, *representational* singularities exist too, for example, the FVK or IVK algorithms exhibit one of the mathematical singularities inherent in any minimal coordinate representation, Sect. 4.1.5.

If one or more degrees of motion freedom disappear in a singularity, a corresponding number of reciprocal screws appear. The *reciprocal wrench space* is at least one-dimensional, or increases in dimension if it was already non-empty outside of the singularity, [148, 251]. The wrenches in the reciprocal wrench space are transmitted from the end-effector to the base completely *passively*, i.e., just by the mechanical structure of the robot without *any* need for joint forces/torques.

7.19.1 Singularities for the 321 structure

The singular positions of the 321 robot structure, Fig. 7.11, 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. (7.17). Hence, it vanishes in the following three cases (Fig. 7.16):

Arm-extended singularity ($q_3 = 0$) ([148] 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.

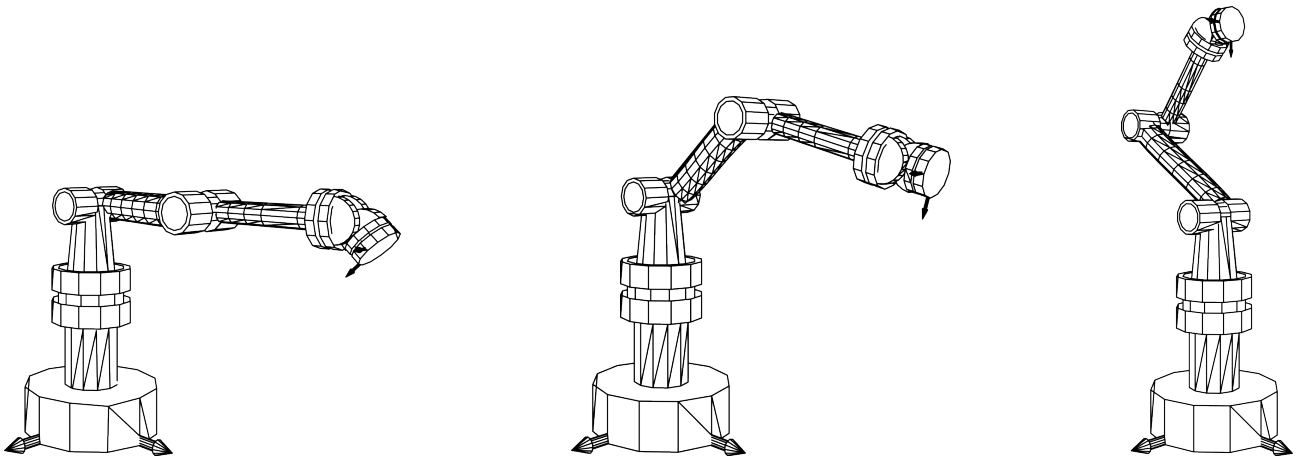

Figure 7.16: The three singular positions for the 6R wrist-partitioned serial robot arm with closed form kinematic solutions: "arm-extended," "wrist-extended" and "wrist-above-shoulder."

Wrist-extended singularity ($q_5 = 0$) ([148] 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, [148].

Wrist-above-shoulder singularity ($d^h = 0$) ([148] 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 segments.

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, [72, 125, 273]. Only special structures, such as the 321 robots, have their singularities coinciding with their configuration borders.

The zero block in the Jacobian matrix for a 321 design, Eq. (7.33), 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 of the wrist, $\det(\boldsymbol{B}) = 0$, and the singularities of the regional structure of the arm, $\det(\boldsymbol{C}) = 0$, [278].

A general kinematic structure has more complicated and less intuitive singularities. The reason why *shoulder offsets*, Fig. 7.4, 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.

7.19.2 Singularities for the 313 structure

Chapter 8

Parallel robots

Chapter 7 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.

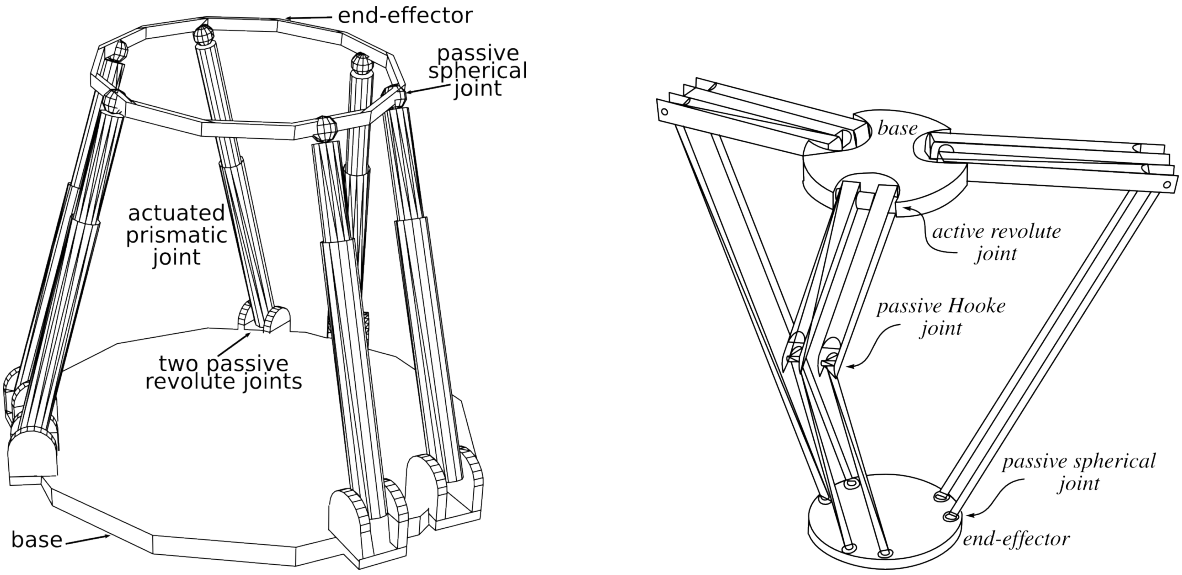

Figure 8.1: Fully parallel robots. Left: General *Stewart-Gough platform*; the actuated joints are prismatic, the passive joints are revolute. Right: *HEXA platform*; all joints are revolute.

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 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, [257].

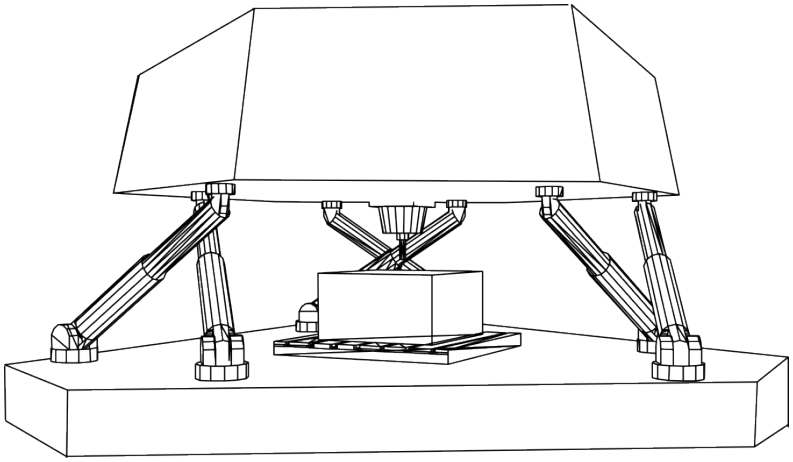

Figure 8.2: Milling machine with a parallel manipulator design (*Variax*, by Gidding & Lewis).

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. This Chapter exploits these dualities by describing the kinematics of parallel robots by the same geometrical concepts used in the serial manipulator case.

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, [6, 87].

8.1 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, [88]. 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, [246]. 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, [160], 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, [202]. This was the successor of the three degrees of freedom *DELTA* robot, [49]. 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 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. 7.1.

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.7.15) 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 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.

8.3 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, [122, 159]. 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, [123].

8.4 Coordinate conventions and transformations

The link frame conventions and transformations defined for serial kinematic chains (Sect. 5.3) 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

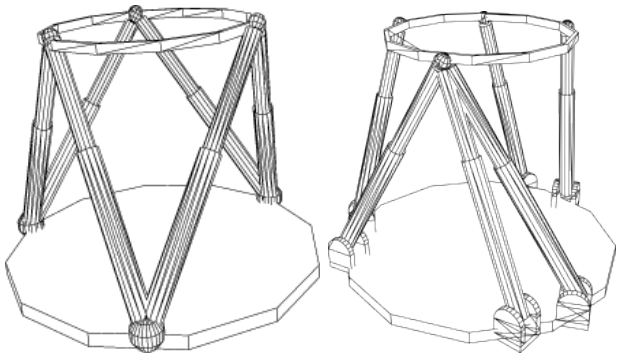

Figure 8.3: Left: 3-3 "Stewart(-Gough)" design. Right: 3-1-1-1 design.

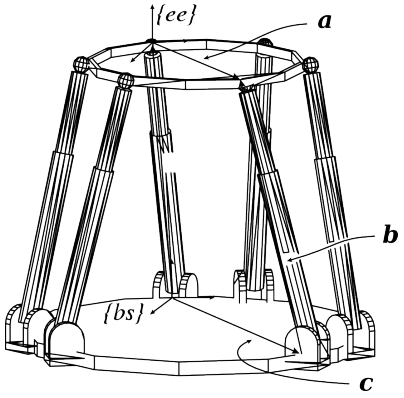

Figure 8.4: Generic kinematic model for a parallel manipulator.

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:

$$\boldsymbol{p}^{bs,i_{bs}} + \boldsymbol{l}_i = \boldsymbol{p}^{bs,ee} + \boldsymbol{p}^{ee,i_{ee}}, \quad \forall i = 1, \ldots, 6. \tag{8.1}$$

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.

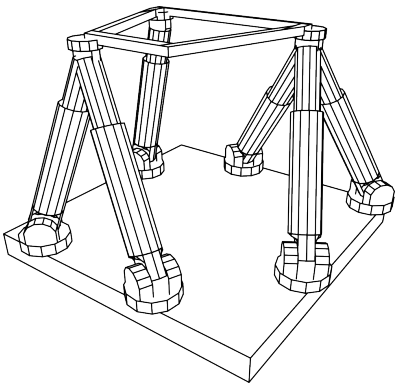

Figure 8.5: 321 parallel structure.

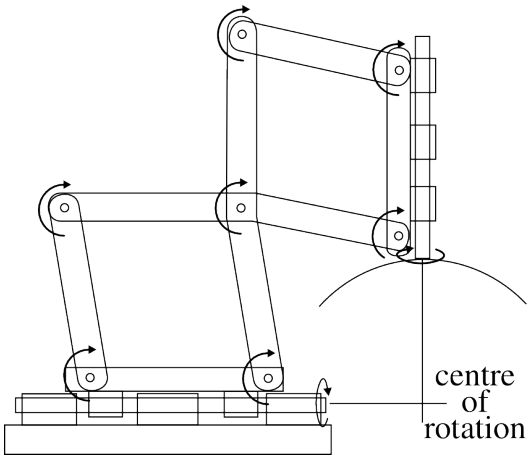

Figure 8.6: The "CMS" spherical joint, [101]. This design allows to connect several links to functionally concentric spherical joints.

8.5 321 kinematic structure

For serial robots, the 321 kinematic structure, Sect. 7.2, 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," [179, 180, 101]. 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, [101], and an alternative design is given in [20].

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.

8.6 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* $\boldsymbol{q} = (q_1 \ \ldots \ q_n)^T$*?* 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 $\boldsymbol{p}bs^{bs,i_{bs}}$ 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_i^b, Fig 8.7) in leg i to the position $\boldsymbol{p}_i$ given by

$$\boldsymbol{p}_i = \boldsymbol{b}_i + {}^i_{bs}\boldsymbol{R}\ \boldsymbol{R}(X, q_i) \begin{pmatrix} 0 & 0 & l_i^b \end{pmatrix}^T. \tag{8.4}$$

${}^i_{bs}\boldsymbol{R}$ 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 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. 3.8):

$$\boldsymbol{R}(X, q_i) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(q_i) & -\sin(q_i) \\ 0 & \sin(q_i) & \cos(q_i) \end{pmatrix}. \tag{8.5}$$

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

$$||\boldsymbol{p}_i - {}_{bs}\boldsymbol{t}_i|| = l_i^t, \tag{8.6}$$

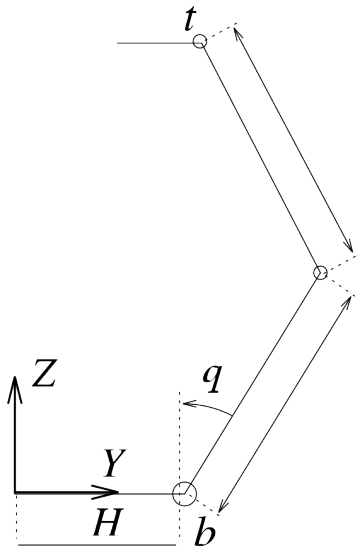

Figure 8.7: One leg in the HEXA design. The joint angle q_i is variable and measured; the lengths l_i^b and l_i^t of the "base" and "top" limbs of each leg are constant; the angles of all other joints are variable but not measured. Note that the joint between l_i^t and l_i^b is a two degrees of freedom universal joint, so that the link l_i^t does not necessarily lie in the plane of the figure.

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.

8.7 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:

$$ \mathbf{w}^{ee} = (\boldsymbol{G}_1\ \ldots\ \boldsymbol{G}_n) \begin{pmatrix} \tau_1 \\ \vdots \\ \tau_n \end{pmatrix} = \boldsymbol{G}(\boldsymbol{T})\boldsymbol{\tau}, \tag{8.8} $$

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.)

8.8 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, [116], and the RUS for the HEXA design, [30, 257]. The partial wrench for all joints in these special chains can be found by inspection.

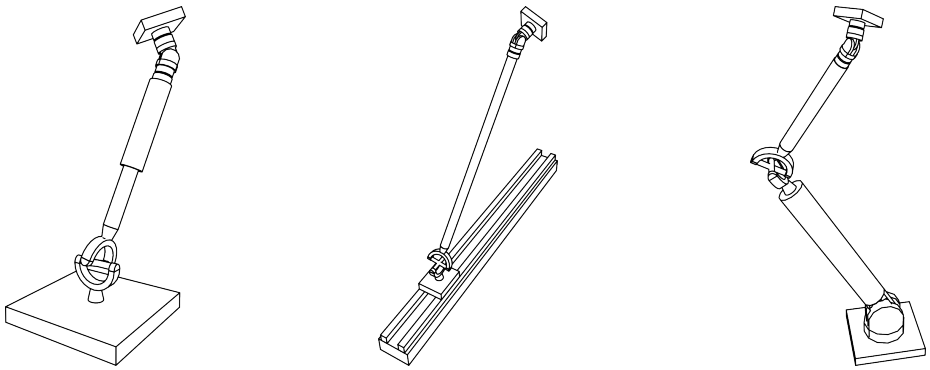

Figure 8.8: UPS, PUS and RUS "legs." "R" stands for *revolute* joint, "P" for *prismatic* joint, "S" for *spherical* joint and "U" for *universal* joint.

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.

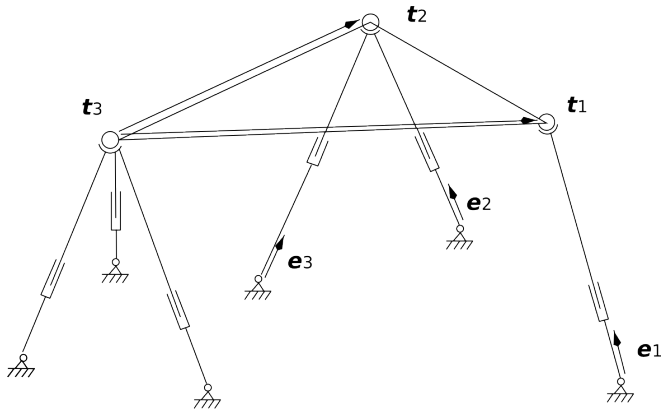

Figure 8.9: 321 structure, with notations. $\boldsymbol{t}_i$ is the point where i legs intersect. $\boldsymbol{e}_i$ is a unit vector along the ith leg.

8.9 Wrench basis for 321 structure

The Jacobian matrix $\boldsymbol{J}$ for the serial 321 structure was derived by inspection (Sect. 7.7), 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.

8.10 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:

$$ \dot{\boldsymbol{q}} = (\boldsymbol{\Delta G})^T \mathbf{t}^{ee}. \tag{8.10} $$

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 Eq. (2.27) 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., ${}_{bs}\boldsymbol{e}^{i}_{z}$. 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

$$\dot{\boldsymbol{l}}_i = ({}_{i}\mathbf{t}^{ee})_6 = \begin{pmatrix} {}_{bs}\boldsymbol{p}^{bs,i_{ee}} \times {}_{bs}\boldsymbol{e}^{i}_{z} \\ {}_{bs}\boldsymbol{e}^{i}_{z} \end{pmatrix}^{T} {}_{bs}\mathbf{t}^{ee} = \left(\boldsymbol{\Delta}\begin{pmatrix} {}_{bs}\boldsymbol{e}^{i}_{z} \\ {}_{bs}\boldsymbol{p}^{bs,i_{ee}} \times {}_{bs}\boldsymbol{e}^{i}_{z} \end{pmatrix}\right)^{T} {}_{bs}\mathbf{t}^{ee}. \tag{8.14}$$

Step 5 The six-vector in Eq. (8.14) between the $\boldsymbol{\Delta}$ 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!).

8.11 Forward position kinematics

The forward position kinematics ("FPK") solves the following problem: *Given the vector of leg positions* $\boldsymbol{q} = (q_1 \ \ldots \ q_n)^T$*, 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, [208]; this reduces to a 16th degree polynomial for the special designs of the Stewart-Gough platform, [91, 181]. The following subsections discuss how to solve the IPK problem *numerically*, but also which designs allow *closed-form* solutions.

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").

Recall Cauchy's Theorem, which, when applied to parallel manipulators with convex base and end-effector designs, implies the following practically important fact: from all possible solutions to the FPK problem, only *one* gives a convex manipulator configuration.

Figure 8.10 shows the convex and a non-convex configuration for the parallel manipulator in Figure 8.3; both configurations have the same leg lengths.

Figure to be made

Figure 8.10: Two solutions for the forward position kinematics of the parallel manipulator in Figure 8.3.

8.12 General parallel structure

The numerical procedure, [67, 250], runs along the same lines as the IPK for a serial robot (Sect. 7.8):

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. 8.6 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:

$$\mathbf{t}_{\Delta,i+1} = (\boldsymbol{\Delta G}_i)^{-T}\Delta\boldsymbol{q}_i. \tag{8.17}$$

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. (4.6).

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) requires a *joint space* start configuration; the FPK for parallel robots, however, requires a *Cartesian space* initial estimate [162], 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.

8.13 FPK by leg length error minimization

The literature contains a second approach to the numerical solution of the FPK. Instead of using the Jacobian transpose equation (8.10) to find a new iteration, the (square of the) error E_{ll} between the desired and current estimate of the leg lengths is *minimised* by a numerical procedure. Hence, the FPK problem is formulated as a minimization procedure derived from Eq. (8.2), [46, 162]:

$$\min_{\boldsymbol{q}} \quad E_{ll}^2 = \sum_{i=1}^{6} \left\{ ({}^{ee}_{bs}\boldsymbol{R}\,\boldsymbol{p}^{ee,i_{ee}} - \boldsymbol{p}^{bs,i_{bs}})^T ({}^{ee}_{bs}\boldsymbol{R}\,\boldsymbol{p}^{ee,i_{ee}} - \boldsymbol{p}^{bs,i_{bs}}) - (q_i)^2 \right\}. \tag{8.19}$$

Solving the FPK problem is then performed by a numerical iteration in which each update step is calculated from a linearization of Eq. (8.19). The leg position vector $\boldsymbol{q}$ is then found as the vector for which Eq. (8.19) reaches its minimum value (which should be zero).

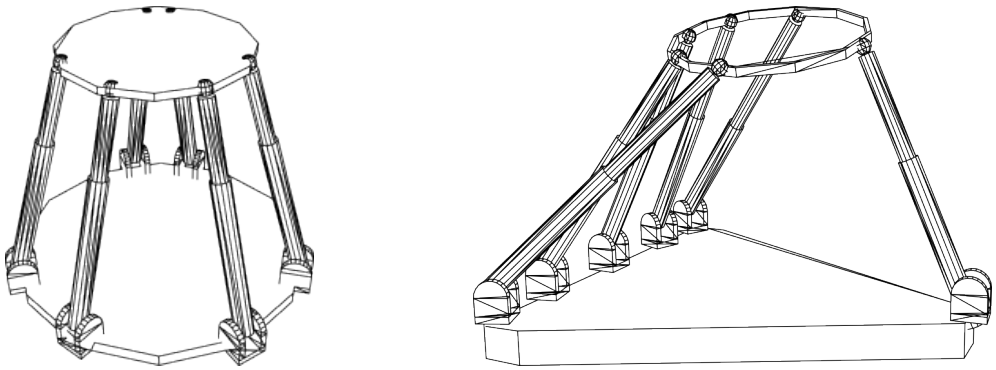

Figure 8.11: Two parallel manipulator designs that allow a closed-form forward kinematics solution: base and end-effector platforms are *similar hexagons* (left), or five leg pivot points are *collinear* (right).

8.14 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, [31, 287]. A special case occurs when both platforms are *similar hexagons* [143, 242, 270], i.e., the pivot points on both platforms lie on polygons that are equal up to a scaling factor, Fig. 8.11. (This design seems attractive at first sight, but it turns out to have a very bad singularity manifold, [32].)

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, [292] (Fig. 8.11).

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. 8.12): 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.

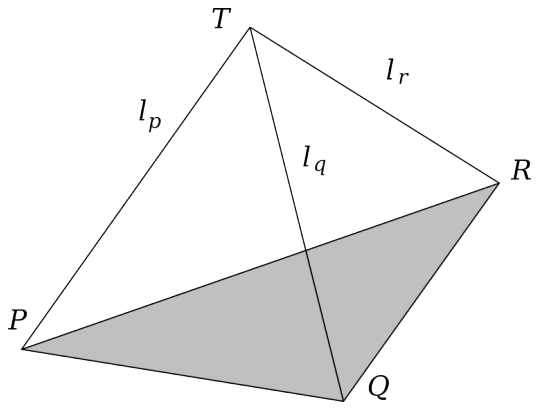

Figure 8.12: Tetrahedral substructure of the 321 structure.

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

$$s_x^2 + s_y^2 + s_z^2 = l_p^2. \tag{8.20}$$

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

$$a_x s_x + a_y s_y + a_z s_z = (l_p^2 + a^2 - l_q^2)/2, \tag{8.21}$$
$$b_x s_x + b_y s_y + b_z s_z = (l_p^2 + b^2 - l_r^2)/2. \tag{8.22}$$

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 (8.21) and (8.22). Equation (8.20) then yields a quadratic equation in s_z, with two solutions. Backsubstitution of the result in (8.21) and (8.22) 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*, [3]. 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.

8.15 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, [16, 102, 126, 143, 163, 175, 191]:

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 8.12. 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.

8.16 Forward velocity kinematics

The forward velocity kinematics ("FVK") of a parallel robot are dual to the inverse velocity kinematics of a serial robot, Sect. 7.11.

8.17 General parallel robots

The obvious numerical technique to solve the FVK inverts the "Jacobian transpose" equation (8.10):

$$\mathbf{t}^{ee} = (\boldsymbol{\Delta G})^{-T}\dot{\boldsymbol{q}}. \tag{8.23}$$

This approach requires the FPK solution, in order to construct the wrench basis $\boldsymbol{G}$.

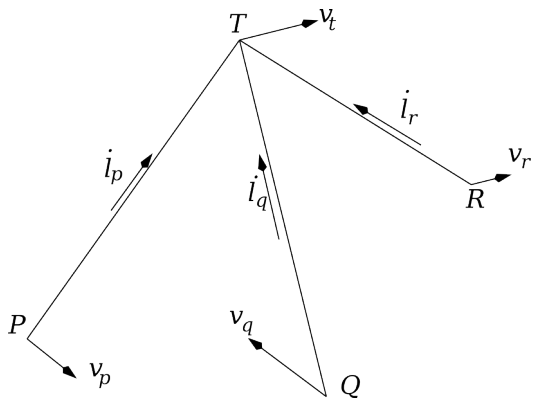

Figure 8.13: Velocities in a tetrahedron of the 321 structure.

8.18 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:

$$\boldsymbol{G} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{C} \end{pmatrix} \Rightarrow \boldsymbol{G}^{-1} = \begin{pmatrix} \boldsymbol{A}^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{C}^{-1} \\ \boldsymbol{0} & \boldsymbol{C}^{-1} \end{pmatrix}. \tag{8.24}$$

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. 8.13. 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:

$$0 = \boldsymbol{e} \cdot (\boldsymbol{t} - \boldsymbol{q}), \tag{8.25}$$
$$0 = \boldsymbol{e} \cdot (\boldsymbol{t} - \boldsymbol{r}), \tag{8.26}$$
$$1 = \boldsymbol{e} \cdot \boldsymbol{e}, \tag{8.27}$$

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 (8.20)–(8.22), 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

$$0 = \frac{\mathrm{d}l^t}{\mathrm{d}t} = \boldsymbol{v}_p \cdot (\boldsymbol{t} - \boldsymbol{p}) - v_t \boldsymbol{e} \cdot (\boldsymbol{t} - \boldsymbol{p}). \tag{8.28}$$

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 [4, 77, 239].

8.19 Singularities

Equation (8.23) 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*.

Two examples of singularities are:

1. The manipulator with similar base and end-effector platforms (Fig. 8.11) is singular if both platforms are *regular hexagons*, i.e., the pivot points lie on radii of a circle which are all 60 degrees apart. This kind of singularity is called an *architectural* singularity, [154], since it is caused by the design of the manipulator, and not by a specific combination of leg lengths.

2. A second kind of singularity is called a *configuration* singularity, [154]. This singularity can only happen for specific values of the leg lengths. One intuitive example occurs when either the base or the end-effector platform are coplanar with one or more legs. In this configuration, the manipulator cannot resist forces orthogonal to the plane of the base or the end-effector.

8.20 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!)

8.21 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. 5.9) 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:

$$\text{Null}\,(\boldsymbol{G}) = \left\{\boldsymbol{\tau}^N \mid \boldsymbol{G}\,\boldsymbol{\tau}^N = \mathbf{0}\right\}. \tag{8.29}$$

   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:

$$\begin{cases} \min\limits_{\boldsymbol{\tau}} \quad P = \dfrac{1}{2}\boldsymbol{\tau}^T\boldsymbol{K}^{-1}\boldsymbol{\tau}, \\ \text{such that} \quad \mathbf{w}^{ee} = \boldsymbol{G}\,\boldsymbol{\tau}. \end{cases} \tag{8.30}$$

   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 $\text{diag}(k_1, \ldots, k_n)$, 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. (5.45)

$$\boldsymbol{\tau} = \boldsymbol{K}\boldsymbol{G}^T\left(\boldsymbol{G}\boldsymbol{K}\boldsymbol{G}^T\right)^{-1}\mathbf{w}^{ee} \tag{8.31}$$

$$= \boldsymbol{G}_K^{\dagger}\mathbf{w}^{ee}. \tag{8.32}$$

   $\boldsymbol{G}_K^{\dagger}$ is the weighted pseudo-inverse, with stiffness matrix $\boldsymbol{K}$ acting as weighting matrix on the space of joint forces.

8.22 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 $\mathbf{t}^{ee} = (\boldsymbol{\Delta G})^{-T}\dot{\boldsymbol{q}}$
FFK	redundancy, singularities $\mathbf{w}^{ee} = (\boldsymbol{\Delta J})^{-T}\boldsymbol{\tau}$	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 $\dot{\boldsymbol{q}} = (\boldsymbol{\Delta G})^{T}\mathbf{t}^{ee}$
IFK	always defined $\boldsymbol{\tau} = (\boldsymbol{\Delta J})^{T}\mathbf{w}^{ee}$	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

Chapter 9

Mobile robot platforms

This Chapter presents the physical properties and mathematical coordinate representations and algorithm for *mobile robots*, i.e., devices such as unicycles, bicycles, cars, and, especially, the mobile devices that have two independently driven wheels on one axle, [197, 245]. This text calls them "*differentially-driven*" robots. "*Caster*" robot is another name, referring to the caster wheel (or "castor" wheel) which helps to keep the robot's balance. A caster wheel is a wheel mounted in a swivel frame, and in daily live also 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* (or a reasonably good approximation of it). Hence, their *configuration* space has two translational and one rotational degree of freedom; the rotation axis is perpendicular to the translations. 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 robots discussed in other 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 autonomously produce 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, except for friction); 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, [84, 184], i.e., they cannot be integrated to give a constraint on the robots' Cartesian *pose*. (The word "holonomic" is built from the Greek words *holos* ("integral") and *nomos* ("law"); the terminology was introduced by Hertz, [108].) In other words, 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; it only *influences the direction* of motion.

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 well defined.

Applications. Mobile robots are used for different purposes than serial and parallel robots, 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 or natural terrain.

Special kinematic designs. Most academic or industrial mobile robots have different kinematic features with respect to real-world trucks or passenger cars, because of reasons of *kinematic simplicity*, *operation under laboratory conditions*, and the requirement of *measuring the position* of the robot. 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. 9.1, [1, 171], or spheres driven by three or more rollers, [78, 266, 274].

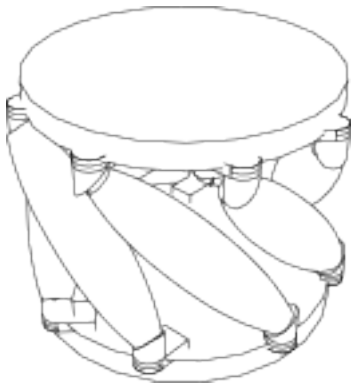

Figure 9.1: Example of a holonomic wheel. The passive "rollers" allow rolling of the wheel in all directions.

Mobile robots are *simpler* than serial and parallel robots since only planar motions are involved. On the other hand, they are more *complex to control* than serial and parallel robots, because of the *nonholonomic* constraints on their instantaneous velocity. At the instantaneous, velocity level, mobile robots can be treated mathematically as a special type of *parallel* robot (i.e., it has different connections to the ground in parallel).

9.1 Coordinate representations

A mobile robot is a rigid body in E^3, 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. 9.2: 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: it gives the nonholonomic constraint gets a simpler *coordinate* expression, and makes the influence of left and right driven wheels symmetric in the *coordinate* equations of motion. However, the kinematic properties presented in the following Sections are independent of the choice of reference frames.

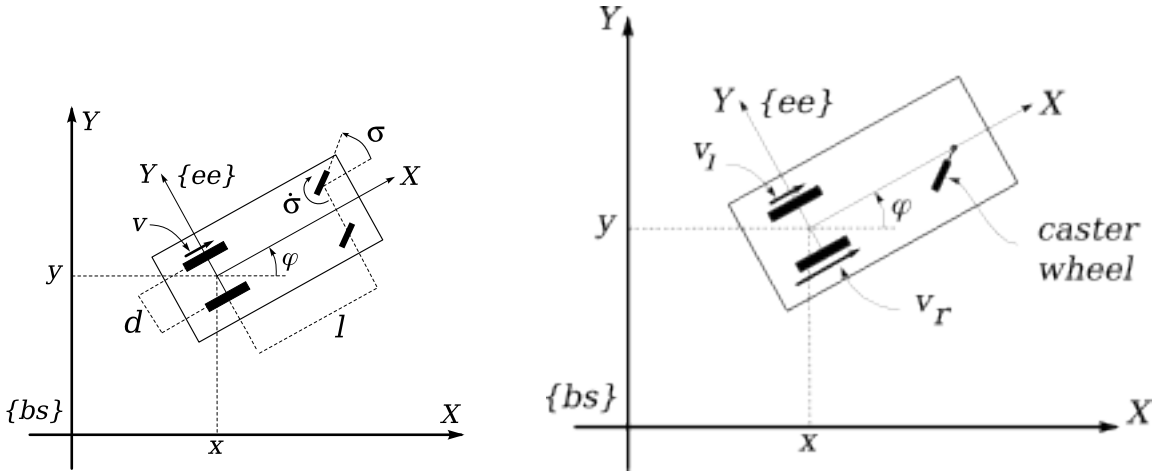

Figure 9.2: Geometric parameters of car-like robot (left) and differentially-driven robot (right).

Pose. The homogeneous transformation matrix is as simple as (Fig. 9.2):

$$ {}^{ee}_{bs}\boldsymbol{T} = \begin{pmatrix} {}^{ee}_{bs}\boldsymbol{R}(Z,\varphi) & \begin{matrix} x \\ y \\ 0 \end{matrix} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix}, \quad \text{with} \quad {}^{ee}_{bs}\boldsymbol{R}(Z,\varphi) = \begin{pmatrix} \cos(\varphi) & -\sin(\varphi) & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{9.1} $$

In the context of mobile robot kinematics, this is simplified to either a 3×3 matrix (denoted by the same symbol ${}^{ee}_{bs}\boldsymbol{T}$) or a finite displacement three-vector twist $\boldsymbol{t}_d$:

$$ {}^{ee}_{bs}\boldsymbol{R} = \begin{pmatrix} \cos(\varphi) & -\sin(\varphi) & x \\ \sin(\varphi) & \cos(\varphi) & y \\ 0 & 0 & 1 \end{pmatrix}, \qquad \boldsymbol{t}_d = \begin{pmatrix} \varphi \\ x \\ y \end{pmatrix}. \tag{9.2} $$

Screw. The instantaneous screw axis (ISA) of Chasles' Theorem in 3D [41] 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 [206] 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 can be represented by a three-vector (denoted by the same symbol):

$$ \boldsymbol{t}_{SE(3)} = \begin{pmatrix} 0 \\ 0 \\ \omega \\ v_x \\ v_y \\ 0 \end{pmatrix} \Rightarrow \boldsymbol{t}_{SE(2)} = \begin{pmatrix} \omega \\ v_x \\ v_y \end{pmatrix}. \tag{9.3} $$

Similarly, the 2D wrench three-vector becomes $\boldsymbol{w} = (f_x \;\; f_y \;\; m)^T$, resulting from putting to zero three *other* elements in the 3D screw:

$$ \boldsymbol{w}_{SE(3)} = \begin{pmatrix} f_x \\ f_y \\ 0 \\ 0 \\ 0 \\ m \end{pmatrix} \Rightarrow \boldsymbol{w}_{SE(2)} = \begin{pmatrix} f_x \\ f_y \\ m \end{pmatrix}. \tag{9.4} $$

A twist transforms with the 3D screw transformation matrix simplified as follows:

$$ \begin{pmatrix} {}_a\omega \\ {}_av_x \\ {}_av_y \end{pmatrix} = {}^b_a\boldsymbol{S}_{\boldsymbol{t}} \begin{pmatrix} {}_b\omega \\ {}_bv_x \\ {}_bv_y \end{pmatrix} \quad \text{with} \quad {}^b_a\boldsymbol{S}_{\boldsymbol{t}} = \begin{pmatrix} 1 & 0 & 0 \\ y & c_\varphi & -s_\varphi \\ -x & s_\varphi & c_\varphi \end{pmatrix}. \tag{9.5} $$

Wrenches transform with the 3D screw transformation matrix simplified as follows:

$$\begin{pmatrix} {}_af_x \\ {}_af_y \\ {}_am \end{pmatrix} = {}^b{}_a\boldsymbol{S}_{\boldsymbol{w}} \begin{pmatrix} {}_bf_x \\ {}_bf_y \\ {}_bm \end{pmatrix} \quad \text{with} \quad {}^b_a\boldsymbol{S}_{\boldsymbol{w}} = \begin{pmatrix} c_\varphi & -s_\varphi & 0 \\ s_\varphi & c_\varphi & 0 \\ -yc_\varphi + xs_\varphi & ys_\varphi + xc_\varphi & 0 \end{pmatrix} \begin{pmatrix} \omega \\ {}_bv_x \\ {}_bv_y \end{pmatrix}. \tag{9.6}$$

The reciprocity between twists and wrenches becomes:

$$\boldsymbol{t}^T \boldsymbol{\Delta} \boldsymbol{w} = 0, \quad \text{with} \quad \boldsymbol{\Delta} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad \text{and} \quad \boldsymbol{\Delta}^{-1} = \boldsymbol{\Delta}^T. \tag{9.7}$$

9.2 Kinematic models

Mobile robots have quite simple mathematical models to describe their instantaneous motion capabilities; especially compared to serial, parallel and humanoid robots. However, this only holds for single mobile robots only, because the modelling does become complex as soon as one begins to add *trailers* to mobile robots. Airport luggage carts are a fine example of such mobile robot trains.

9.2.1 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 are most often 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, Figures 9.3–9.4 show 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. 9.3).

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.

9.2.2 Centre of rotation

Figures 9.5–9.6 show 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 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*, [245], or *instantaneous rotation radius* r^{ir}. Figures 9.5–9.6 and some simple trigonometry show that

$$r^{ir} = \begin{cases} \dfrac{l}{\tan(\sigma)}, & \text{for a car-like robot,} \\ \\ \dfrac{d}{2}\dfrac{v_r + v_l}{v_r - v_l}, & \text{for a differentially-driven robot.} \end{cases} \tag{9.8}$$

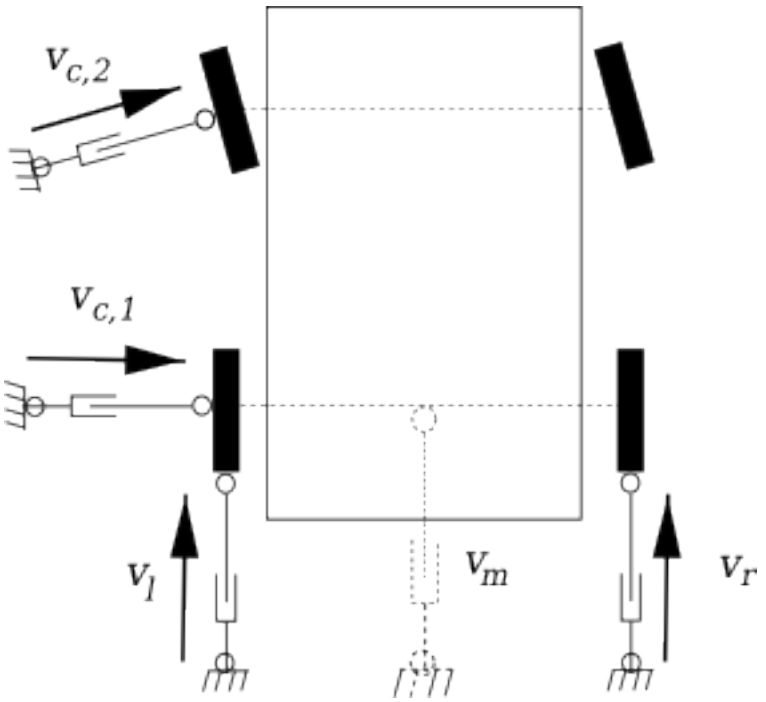

Figure 9.3: Instantaneously equivalent parallel manipulator models for car-like robot.

with l the *wheelbase* of the car-like robot, [245], (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. 9.2).
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. In practice, 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.

9.2.3 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. 9.7). 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, [35]. 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.
(TODO: give the details.)

Finally, note that the system truck-and-trailer becomes even more constrained (i.e., more difficult *to control* to a desired configuration) 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.

9.3 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 (presented in the Chapter on *Parallel Robots*) uses the wrench basis $\boldsymbol{G}$ of the equivalent parallel manipulator.

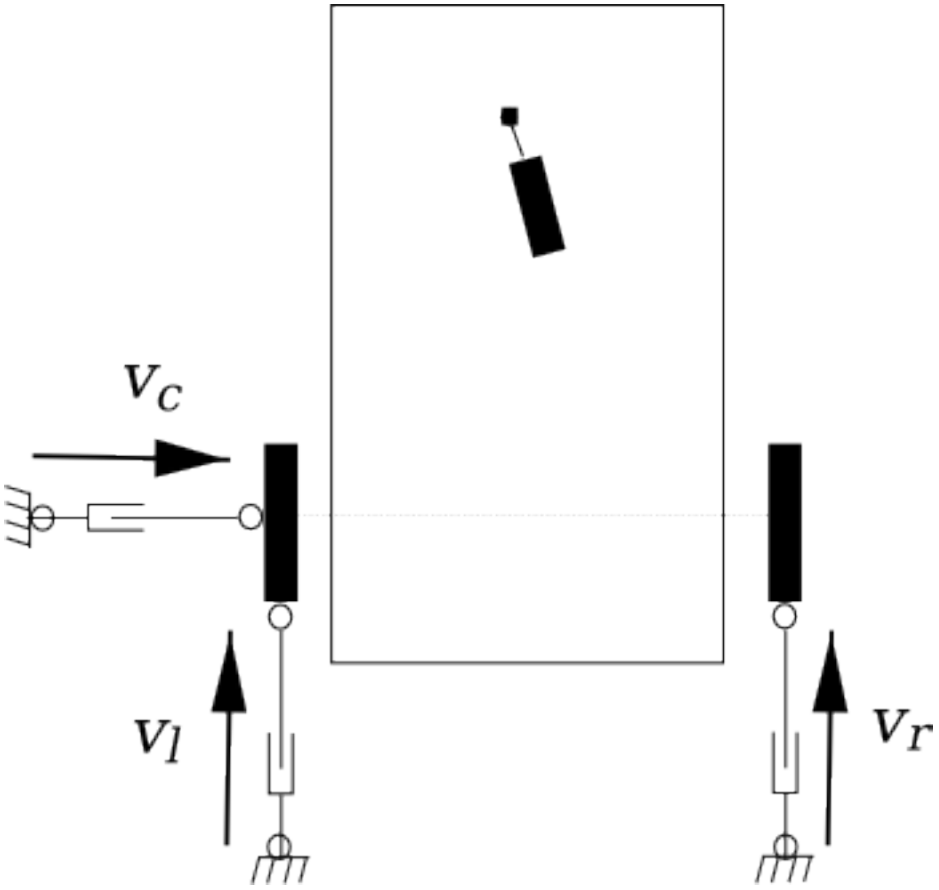

Figure 9.4: Instantaneously equivalent parallel manipulator models for differentially-driven robot.

Since differentially-driven robots and car-like robots have different parallel manipulator models, their wrench bases are different too:

$$\boldsymbol{w}^{ee} = {}_{cast}\boldsymbol{G} \begin{pmatrix} f_r \\ f_l \\ f_c \end{pmatrix}, \qquad \boldsymbol{w}^{ee} = {}_{car}\boldsymbol{G} \begin{pmatrix} f_m \\ f_{c,1} \\ f_{c,2} \end{pmatrix}. \tag{9.9}$$

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 $\boldsymbol{w}^{ee}$ 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 $\boldsymbol{w}^{ee}$ in the constraint directions. The wrench basis $\boldsymbol{G}$ consists of the *partial wrench* of the actuated prismatic joints. 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 9.2) 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:

$${}_{icr}\boldsymbol{G}_{car} = \begin{pmatrix} 1 & 0 & \cos(\sigma) \\ 0 & -1 & -\sin(\sigma) \\ r^{ir} & 0 & 0 \end{pmatrix}, \tag{9.11}$$

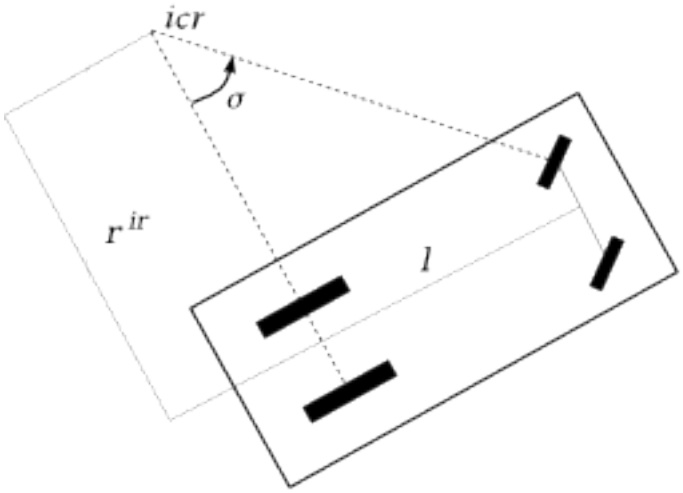

Figure 9.5: Instantaneous centre of rotation (icr) for a car-like robot.

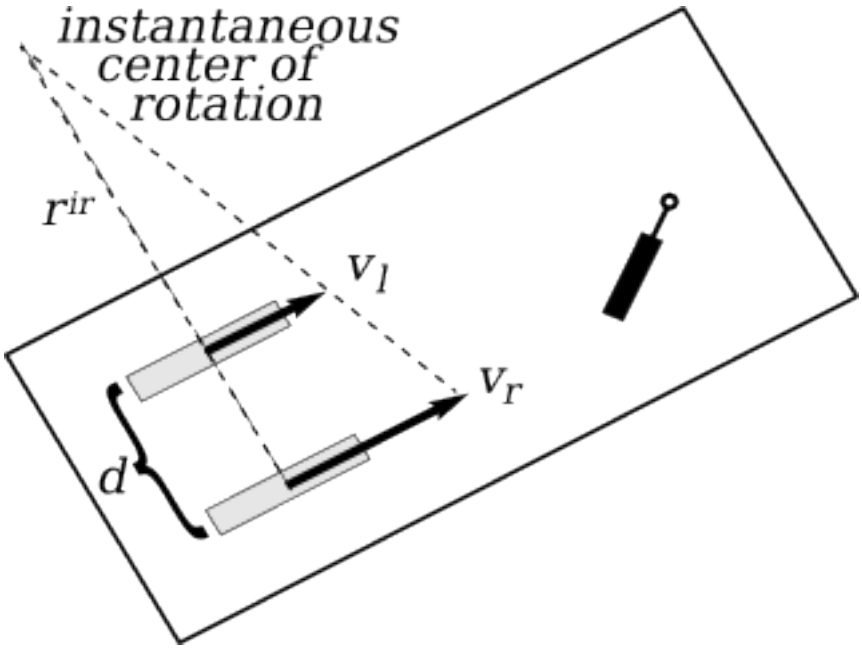

Figure 9.6: Instantaneous centre of rotation for a differentially-driven robot.

with r^{ir} the distance to the instantaneous centre of rotation, Eq. (9.8). Pre-multiplication with the screw transformation matrix ${}^{icr}_{ee}\boldsymbol{S}_{\boldsymbol{w}}$ gives the expression in the end-effector frame $\{ee\}$:

$$ {}_{ee}\boldsymbol{G}_{car} = \begin{pmatrix} 1 & 0 & c_\sigma \\ 0 & -1 & -s_\sigma \\ -r^{ir} & 0 & 1 \end{pmatrix} \; {}_{icr}\boldsymbol{G}_{car} = \begin{pmatrix} 1 & 0 & c_\sigma \\ 0 & -1 & -s_\sigma \\ 0 & 0 & -r^{ir} c_\sigma \end{pmatrix}. \tag{9.12} $$

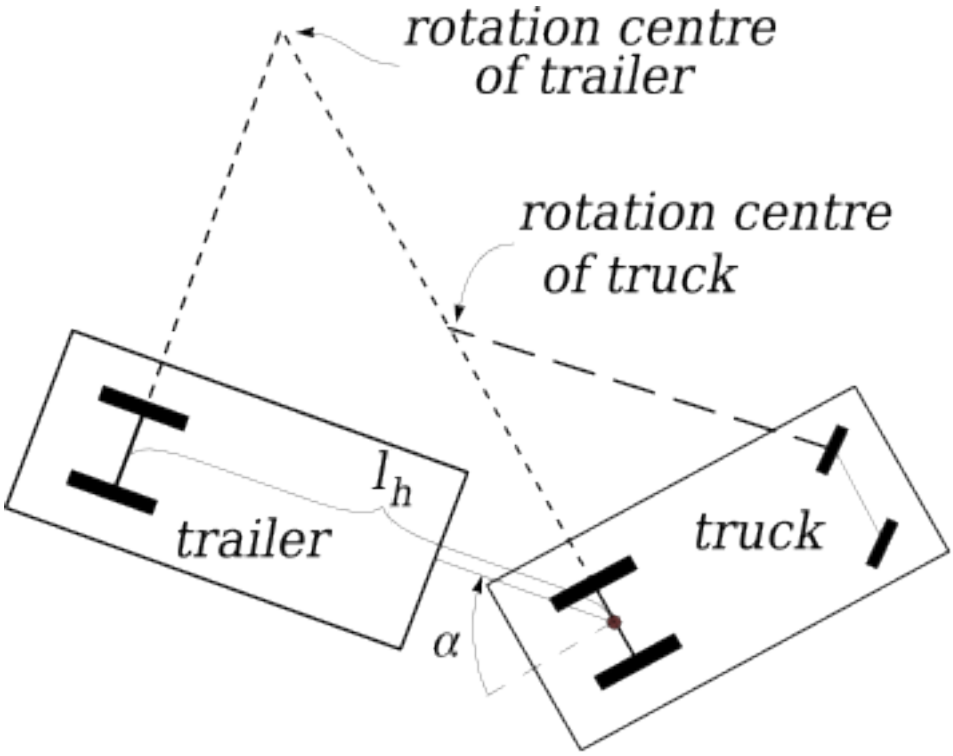

Figure 9.7: Instantaneous rotation centres of truck with trailer.

9.4 Inverse velocity kinematics

Again using the equivalent parallel robot, the IVK for the example of the differentially-driven robot corresponds to:

$$\begin{pmatrix} v_r \\ v_l \\ v_c \end{pmatrix} = (\boldsymbol{\Delta G}_{dd})^T \boldsymbol{t}^{ee}, \quad \text{and} \quad \begin{pmatrix} v_m \\ v_{c,1} \\ v_{c,2} \end{pmatrix} = (\boldsymbol{\Delta G}_{car})^T \boldsymbol{t}^{ee}. \tag{9.13}$$

v_r and v_l are the velocities of the right and left wheels (Fig. 9.3); 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 $\boldsymbol{t}^{ee}$ 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* (see the Chapter on *Serial Robots*) to project the nominally desired end-effector twist $\boldsymbol{t}^{ee}$ 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, [128].

2. External sensors can produce *measured* end-effector twists $\boldsymbol{t}^{ee}$. Using the IVK in Eq. (9.13) then yields the corresponding wheel velocities *and* the transversal slip velocity.

9.5 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 $(\varphi\, x\, y)^T$ of the robot, what is then its corresponding twist $\boldsymbol{t}^{ee}$?"*

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 $(\varphi\, x\, y)^T$ of the robot, what is then its corresponding twist $\boldsymbol{t}^{ee}$?"*

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. 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. For example, in the differentially-driven robot case:

$$\boldsymbol{t}^{ee} = (\boldsymbol{\Delta G})^{-T} \begin{pmatrix} v_r \\ v_l \\ v_c = 0 \end{pmatrix}, \tag{9.15}$$

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:

$$(\boldsymbol{\Delta}\, \boldsymbol{G}_{dd})^{-1} = \begin{pmatrix} \frac{1}{2d} & \frac{1}{2} & 0 \\ -\frac{1}{2d} & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \text{and} \quad (\boldsymbol{\Delta}\, \boldsymbol{G}_{car})^{-1} = \begin{pmatrix} \frac{1}{r^{ir}} & 1 & 0 \\ \frac{\tan(\varphi)}{r^{ir}} & 0 & -1 \\ -\frac{1}{r^{ir}\cos(\varphi)} & 0 & 0 \end{pmatrix}. \tag{9.16}$$

So the FVK with respect to the midframe on the rear axle are given by:

$$\boldsymbol{t}^{ee}_{dd} = \begin{pmatrix} \omega \\ v_x \\ v_y \end{pmatrix} = \begin{pmatrix} \frac{v_r - v_l}{2d} \\ \frac{v_r + v_l}{2} \\ 0 \end{pmatrix}, \quad \text{and} \quad \boldsymbol{t}^{ee}_{car} = \begin{pmatrix} \frac{v_m}{r^{ir}} \\ v_m \\ 0 \end{pmatrix}. \tag{9.17}$$

These FVK relationships could of course also be derived by inspection. The forward velocity kinematics are straightforward, but subject to inaccurate modelling approximations. The inverse velocity kinematics are, in general, not uniquely defined.

9.6 Forward position kinematics—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), [245]. (The Chinese invented a mechanical *hodometer* already in about 265 AD!) Contrary to the trivial case of (holonomic) serial, parallel or humanoid robots, dead reckoning for (nonholonomic) mobile robots *must* be performed by integration of velocity equations such as Eq. (9.17). For example, the dead reckoning for a differentially-driven robot solves the following set of integral equations:

$$ {}_{bs}\boldsymbol{p} = \begin{pmatrix} x(t) \\ y(t) \\ \varphi(t) \end{pmatrix} = \begin{pmatrix} x_0 + \frac{1}{2}\int_0^t \cos(\varphi(t))(r_r\dot{q}_r + r_l\dot{q}_l)\,dt \\ y_0 + \frac{1}{2}\int_0^t \sin(\varphi(t))(r_r\dot{q}_r + r_l\dot{q}_l)\,dt \\ \varphi_0 + \frac{1}{d}\int_0^t (r_r\dot{q}_r - r_l\dot{q}_l)\,dt \end{pmatrix}. \tag{9.18} $$

Any integration scheme will experience *drift*, due to numerical round-off errors. In the case of mobile robots, 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 Inverse position kinematics

In principle, the inverse position kinematics for a mobile robot would have to solve the following problem:

> *Given a desired pose $(\varphi\, x\, y)^T$ 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 to *robustly* bring a mobile robot to its desired position from any given start position, [26, 27, 197, 227].

9.8 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 Section presents the basic *motion operators* for mobile robots, [34, 185].

9.8.1 Differentially-driven robots

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!

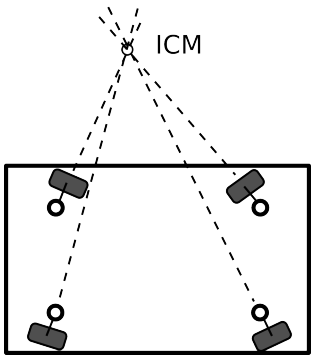

Figure 9.8: Instantaneous rotation centre of redundantly actuated mobile platform.

9.8.2 Car-like robots

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.

9.9 Redundantly actuated mobile platforms

(TODO: . . .)

9.10 Dynamics

(TODO: . . .)

Chapter 10

Mobile manipulators

Chapter 7 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

Bibliography

[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] G. S. Aksenov, D. K. Voronetskaya, and V. N. Fomin. A construction of program movements of a manipulator with the aid of a computer. *Eng. Cybern.*, 16(4):40–45, 1978.

[3] 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.

[4] 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.

[5] J. Angeles. *Rational Kinematics*. Springer, 1988.

[6] J. Angeles. *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms*. Mechanical Engineering Series. Springer-Verlag, 1997.

[7] P. Appell. Sur une forme générale des équations de la dynamique et sur le principe de Gauss. *J. für die Reine und Angewandte Mathematik*, 122:205–208, 1900.

[8] P. Appell. *Traité de Mécanique Rationale*. Paris, 3rd edition, 1911.

[9] V. I. Arnold, K. Vogtmann, and A. Weinstein. *Mathematical Methods of Classical Mechanics*, volume 60 of *Graduate Texts in Mathematics*. Springer, New York, NY, 2nd edition, 1989.

[10] J. Baillieul. Kinematic programming alternatives for redundant manipulators. In *Int. Conf. Robotics and Automation*, pages 722–728, St. Louis, MS, 1985.

[11] J. Baillieul. Avoiding obstacles and resolving kinematic redundancy. In *Int. Conf. Robotics and Automation*, pages 1698–1704, San Fransisco, CA, 1986.

[12] D. R. Baker. Some topological problems in robotics. *The Mathematical Intelligencer*, 12(1):66–76, 1990.

[13] D. R. Baker and C. W. Wampler II. On the inverse kinematics of redundant manipulators. *Int. J. Robotics Research*, 7(2):3–21, 1988.

[14] A. Balestrino, G. G. De Maria, and L. Sciavicco. Robust control of robotic manipulators. In *Proc. 9th IFAC*, pages 2435–2440, Budapest, Hungary, 1984.

[15] R. S. Ball. *Theory of screws: a study in the dynamics of a rigid body*. Hodges, Foster and Co, Dublin, Ireland, 1876. Reprinted 1998, by Cambridge University Press.

[16] L. Baron and J. Angeles. The decoupling of the direct kinematics of parallel manipulators using redundant sensors. In *Int. Conf. Robotics and Automation*, pages 974–979, San Diego, CA, 1994.

[17] J. S. Bay. Geometry and prediction of drift-free trajectories for redundant machines under pseudoinverse control. *Int. J. Robotics Research*, 11(1):41–52, 1992.

[18] A. Ben-Israel and T. N. E. Greville. *Generalized Inverses: Theory and Applications*. Robert E. Krieger Publishing Company, Huntington, NY, reprinted edition, 1980.

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

[20] 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.

[21] R. Beyer. *Technische Raumkinematik*. Springer, Berlin, Deutschland, 1963.

[22] 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.

[23] 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.

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

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

[26] 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, Dordrecht, The Netherlands, 1981. Reidel.

[27] R. W. Brockett. Control theory and singular riemannian geometry. In *New Directions in Applied Mathematics*, pages 11–27. Springer, New York, NY, 1982.

[28] R. W. Brockett. Robotic manipulators and the product of exponentials formula. In I. P. A. Fuhrman, editor, *Mathematical theory of networks and systems*, pages 120–129. Springer, New York, NY, 1984. Lecture notes in control and information sciences, Vol. 58.

[29] H. Bruyninckx. Some invariance problems in robotics. Technical Report 91P04, Dept. Mech. Eng., Katholieke Univ. Leuven, Belgium, 1991.

[30] H. Bruyninckx. The 321-HEXA: A fully-parallel manipulator with closed-form position and velocity kinematics. In *Int. Conf. Robotics and Automation*, pages 2657–2662, Albuquerque, NM, 1997.

[31] 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.

[32] H. Bruyninckx and J. De Schutter. Comments on "Closed Form Forward Kinematics Solution to a Class of Hexapod Robots". *IEEE Trans. Rob. Automation*, 15(4):788–789, 1999.

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

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

[35] 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.

[36] S. R. Buss and J.-S. Kim. Selectively damped least squares for inverse kinematics. *Journal of Graphics Tools*, 10(3):37–49, 2005.

[37] 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.

[38] S. L. Campbell and C. D. Meyer, Jr. *Generalized Inverses of Linear Transformations*. Dover, 1991.

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

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

[41] 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.

[42] P. L. Chebyshev. Théorie des mécanismes connus sous le nom de parallélogrammes. *Mémoires présentés à l'Académie Iméríale des sciences de St-Pétersbourg par divers savants*, 7:539–568, 1854. See also Oeuvres de Pafnutii Lvovich Tchebychef, Volume 1, 111–143, Chelsea, New York, 1961.

[43] P. L. Chebyshev. Sur les parallélogrammes. In A. Markoff and N. Sonin, editors, *Oeuvres de Pafnutii Lvovich Tchebychef*, volume 1, pages 83–106. Chelsea Publishing Company, New York, New York, 1961. Russian original 1869.

[44] F.-T. Cheng, T.-H. Chen, and Y.-Y. Sun. Resolving manipulator redundancy under inequality constraints. *IEEE Trans. Rob. Automation*, 10(1):65–71, 1994.

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

[46] K. C. Cheok, J. L. Overholt, and R. R. Beck. Exact methods for determining the kinematics of a Stewart platform using additional displacement sensors. *J. Robotic Systems*, 10(5):689–707, 1993.

[47] P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Siciliano. Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy. *Int. J. Robotics Research*, pages 410–425, 1991.

[48] 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.

[49] R. Clavel. Delta, a fast robot with parallel geometry. In *Int. Symp. Industrial Robots*, pages 91–100, Lausanne, Switzerland, 1988.

[50] 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.

[51] A. Codourey. Dynamic modelling and mass matrix evaluation of the DELTA parallel robot for axes decoupling control. In *Proc. IEEE/RSJ Int. Conf. Int. Robots and Systems*, pages 1211–1218, Osaka, Japan, 1996.

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

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

[54] R. Courant and D. Hilbert. *Methoden der Mathematischen Physik I*. Springer, 1968.

[55] R. Courant and D. Hilbert. *Methods of mathematical physics*. Interscience, New York, NY, 1970.

[56] J. J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley, Reading, MA, 1986.

[57] 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.

[58] T. H. Davies. Mechanical networks–I Passivity and redundancy. *Mechanism and Machine Theory*, 18(2):95–101, 1983.

[59] G. G. de Coriolis. Mémoire sur les équations du mouvement relatif des systèmes des corps. *J. de l'Ecole Polytechnique*, 14(24):142–154, 1835.

[60] P. de Fermat. ?? In *Varia opera mathematica*. Culture et civilisation, Brussels, Belgium, 1679.

[61] J. G. de Jalón and E. Bayo. *Kinematic and Dynamic Simulation of Multibody Systems—The Real Time Challenge*. Springer, 1993.

[62] A. De Luca and G. Oriolo. Nonholonomic behavior in redundant robots under kinematic control. *IEEE Trans. Rob. Automation*, 13(5):776–782, 1997.

[63] P. L. M. de Maupertuis. Essai de cosmologie. In *Oeuvres, Tome I.* Olms, Hildesheim, Germany, 1759.

[64] P. L. M. de Maupertuis. Accord de différentes lois de la nature. In *Oeuvres, Tome IV.* Olms, Hildesheim, Germany, 1768.

[65] 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.

[66] A. S. Deo and I. D. Walker. Minimum effort inverse kinematics for redundant manipulators. *IEEE Trans. Rob. Automation*, 13(5):767–775, 1997.

[67] 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.

[68] K. L. Doty. Tabulation of the symbolic midframe Jacobian of a robot manipulator. *Int. J. Robotics Research*, 6(4):85–97, 1987.

[69] 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.

[70] K. L. Doty, C. Melchiorri, E. M. Schwartz, and C. Bonivento. Robot manipulability. *IEEE Trans. Rob. Automation*, 11(3):462–468, 1995.

[71] 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.

[72] 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.

[73] 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, Switserland, 1952.

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

[75] R. Featherstone. Position and velocity transformations between robot end-effector coordinates and joint angles. *Int. J. Robotics Research*, 2(2):35–45, 1983.

[76] R. Featherstone. *Robot dynamics algorithms.* Kluwer, Boston, MA, 1987.

[77] 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.

[78] L. Ferrière, B. Raucent, and G. Campion. Design of omnimobile robot wheels. In *Int. Conf. Robotics and Automation*, pages 3664–3670, Minneapolis, MN, 1996.

[79] G. R. Fowles. *Analytical Mechanics.* Holt, Rinehart and Winston, New York, NY, 3rd edition, 1977.

[80] K. F. Gauss. Uber ein neues allgemeines Grundgesatz der Mechanik. *J. für die Reine und Angewandte Mathematik*, 4:232–235, 1829.

[81] J. W. Gibbs. On the fundamental formulae of dynamics. *American Journal of Mathematics*, 2:49–64, 1879.

[82] 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.

[83] G. Gogu. Chebychev-Grübler-Kutzbach's criterion for mobility calculation of multi-loop mechanisms revisited via theory of linear transformations. *European Journal of Mechanics - A/Solids*, 24(3):427–441, 2005.

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

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

[86] C. Gosselin and J. Angeles. Singularity analysis of closed-loop kinematic chains. *IEEE Trans. Rob. Automation*, 6(3):281–290, 1990.

[87] 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.

[88] V. E. Gough and S. G. Whitehall. Universal tyre test machine. In *Proc. 9th Int. Tech. Congress FISITA*, pages 117–137, 1962.

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

[90] T. N. E. Greville. The pseudo-inverse of a rectangular or singular matrix and its application to the solution of systems of linear equations. *SIAM Review*, 1(1):38–43, 1959.

[91] M. Griffis and J. Duffy. A forward displacement analysis of a class of Stewart platforms. *J. Robotic Systems*, 6(6):703–720, 1989.

[92] M. F. Grübler. Allgemeine Eigenschaften der Zwangläufigen Ebenen kinematische Kette: I. *Zivilingenieur*, 29:167–200, 1883.

[93] M. F. Grübler. Allgemeine Eigenschaften der Zwangläufigen Ebenen kinematische Kette: II. *Verhandlungen des Verein zur Beförderung des Gewerbefleisses*, 64:179–223, 1885.

[94] M. F. Grübler. *Lehrbuch der technischen Mechanik*. Springer, Berlin, Germany, 1921.

[95] Y.-L. Gu and J.-S. Ho. A unified representation and its applications to robotic control. In *Int. Conf. Robotics and Automation*, pages 98–103, Cincinnati, OH, 1990.

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

[97] W. R. Hamilton. On a general method in dynamics. *Philosophical Trans. of the Royal Society*, (II):247–308, 1834. Reprinted in Hamilton: The Mathematical Papers, Cambridge University Press, 1940.

[98] W. R. Hamilton. Second essay on a general method in dynamics. *Phil. Trans. of the R. Soc.*, (I):95–144, 1835. Reprinted in Hamilton: The Mathematical Papers, Cambridge University Press, 1940.

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

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

[101] G. J. Hamlin. *Tetrobot: A Modular System for Reconfigurable Parallel Robots*. PhD thesis, Rensselaer Polytechnic Institute, Troy, NY, 1996.

[102] 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.

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

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

[105] S. Hayati, K. Tso, and G. Roston. Robot geometry calibration. In *Int. Conf. Robotics and Automation*, pages 947–951, Philadelphia, PA, 1988.

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

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

[108] H. Hertz. *Die Prinzipien der Mechanik in neuem Zusammenhange dargestellt*. Barth, Leipzig, Germany, 1894. Reprinted in "The principles of mechanics," Dover, 1956.

[109] H. Hertz. *Die Prinzipien der Mechanik. Einleiting*. Geest und Portig, Leipzig, Germany, 1984. With annotations by Josef Kuczera.

[110] 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.

[111] A. E. Hoerl. Application of ridge analysis to regression problems. *Chemical Engineering Progress*, 58(3):54–59, 1962.

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

[113] J. M. Hollerbach. Optimum kinematic design for a seven degree of freedom manipulator. In Hanafusa and Inoue, editors, *Robotics Research: The Second Int. Symposium*, pages 215–222. MIT Press, Cambridge, MA, 1985.

[114] J. M. Hollerbach and G. Sahar. Wrist-partitioned, inverse kinematic accelerations and manipulator dynamics. *Int. J. Robotics Research*, 2(4):61–76, 1983.

[115] J. M. Hollerbach and K. C. Suh. Redundancy resolution of manipulators through torque optimization. In *Int. Conf. Robotics and Automation*, pages 1016–1021, St. Louis, MS, 1985.

[116] M. Honegger, A. Codourey, and E. Burdet. Adaptive control of the Hexaglide, a 6 dof parallel manipulator. In *Int. Conf. Robotics and Automation*, pages 543–548, Albuquerque, NM, 1997.

[117] W. W. Hooker. A set of r dynamical attitude equations for an arbitrary n-body satellite having r rotational degrees of freedom. *AIAA J.*, 8(7):1205–1207, 1970.

[118] 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.

[119] 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.

[120] K. H. Hunt. Robot kinematics—A compact analytic inverse solution for velocities. *Trans. ASME J. Mech. Transm. Automation Design*, 109:42–49, 1987.

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

[122] K. H. Hunt and P. R. McAree. The octahedral manipulator: Geometry and mobility. *Int. J. Robotics Research*, 17(8):868–885, 1998.

[123] 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.

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

[125] 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.

[126] H. Inoue, Y. Tsusaka, and T. Fukuizumi. Parallel manipulator. In *Third Int. Symposium on Robotics Research*, pages 321–327, Gouvieux, France, 1985.

[127] Z. Ji. Dynamics decomposition for Stewart platforms. *Trans. ASME J. Mech. Design*, 116:67–69, 1994.

[128] Z.-P. Jiang and H. Nijmeijer. Tracking control of mobile robots: A case study in backstepping. *Automatica*, 33(7):1393–1399, 1997.

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

[130] 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.

[131] 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.

[132] O. Khatib, L. Sentis, and J.-H. Park. A unified framework for whole-body humanoid robot control with multiple constraints and contacts. In *Eur. Rob. Sym.*, pages 303–312, Prague, Czech Republic, March 2008.

[133] O. Khatib, L. Sentis, J.-H. Park, and J. Warren. Whole-body dynamic behavior and control of human-like robots. *International Journal of Humanoid Robotics*, 1(1):29–43, March 2004.

[134] J. Kieffer. Differential analysis of bifurcations and isolated singularities for robots and mechanisms. *IEEE Trans. Rob. Automation*, 10(1):1–10, 1994.

[135] C. A. Klein and B. Blaho. Dexterity measures for the design and control of kinematically redundant manipulators. *Int. J. Robotics Research*, 6(2):72–83, 1987.

[136] C. A. Klein and C. H. Huang. Review of pseudoinverse control for use with kinematically redundant manipulators. *IEEE Trans. on Systems, Man, and Cybernetics*, 13:245–250, 1983.

[137] C. A. Klein and K.-B. Kee. The nature of drift in pseudoinverse control of kinematically redundant manipulators. *IEEE Trans. Rob. Automation*, 5(2):231–234, 1989.

[138] V. Kumar and K. J. Waldron. The workspace of a mechanical manipulator. *Trans. ASME J. Mech. Design*, 103:665–672, 1981.

[139] K. Kutzbach. Mechanische Leitungsverzweigung. Ihre Gesetze und Anwendungen. *Maschinenbau: der Betrieb/Wirtschaftlicher Teil*, 8:710–716, 1929.

[140] J. L. Lagrange. Mécanique analytique. In J.-A. Serret, editor, *Oeuvres*. Gauthier-Villars, Paris, France, 1867.

[141] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Prentice Hall, 1974.

[142] 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.

[143] H.-Y. Lee and B. Roth. A closed-form solution of the forward displacement analysis of a class of in-parallel mechanisms. In *Int. Conf. Robotics and Automation*, pages 720–724, Atlanta, GA, 1993.

[144] K. Levenberg. A method for the solution of certain problems in least squares. *Quart. Appl. Math.*, 2:164–168, 1944.

[145] A. Liegeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-7(12):868–871, 1977.

[146] 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.

[147] J. Lloyd. *Robot Trajectory Generation for Paths with Kinematic Singularities*. PhD thesis, McGill University, Montreal, Canada, 1995.

[148] 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.

[149] G. L. Long, R. P. Paul, and W. D. Fisher. The Hamilton wrist: a four-revolute-joint spherical wrist without singularities. In *Int. Conf. Robotics and Automation*, pages 902–907, Scottsdale, AZ, 1989.

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

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

[152] 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.

[153] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul. On-line computational scheme for mechanical manipulators. *Trans. ASME J. Dyn. Systems Meas. Control*, 102(2):69–76, 1980.

[154] O. Ma and J. Angeles. Architecture singularities of platform manipulators. In *Int. Conf. Robotics and Automation*, pages 1542–1547, Sacramento, CA, 1991.

[155] 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.

[156] S. MacLane and G. Birkhoff. *Algebra*. MacMillan, New York, NY, 2nd edition, 1979.

[157] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math.*, 11:431–441, 1963.

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

[159] 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.

[160] 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.

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

[162] J.-P. Merlet. Parallel manipulators: Part I: Theory; design, kinematics, dynamics and control. Technical Report 646, INRIA, Sophia Antipolis, France, 1987.

[163] J.-P. Merlet. Closed-form resolution of the direct kinematics of parallel manipulators using extra sensor data. In *Int. Conf. Robotics and Automation*, pages 200–204, Atlanta, GA, 1993.

[164] 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.

[165] 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.

[166] E. H. Moore. On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society*, 26:389 and 394–395, 1920.

[167] E. H. Moore and R. W. Barnard. *General Analysis*. American philosophical society, 1935.

[168] B. W. Mooring, Z. S. Roth, and M. R. Driels. *Fundamentals of Manipulator Calibration*. Wiley, New York, NY, 1991.

[169] P. M. Morse and H. Feshbach. *Methods of theoretical physics*. McGraw-Hill, New York, NY, 1953.

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

[171] P. F. Muir and C. P. Neuman. Resolved motion rate and resolved acceleration servo-control of wheeled mobile robots. In *Int. Conf. Robotics and Automation*, pages 1133–1140, Cincinnati, OH, november 2002. To appear in *Probabilistic Graphical Models*, M. Jordan.

[172] R. Mukherjee and Y. Nakamura. Formulation and efficient computation of inverse dynamics of space robots. *IEEE Trans. Rob. Automation*, 8(3):400–406, 1992.

[173] R. M. Murray, Z. Li, and S. S. Sastry. *A mathematical introduction to robotic manipulation.* CRC Press, Boca Raton, FL, 1994.

[174] F. A. Mussa-Ivaldi and N. Hogan. Integrable solutions of kinematic redundancy via impedance control. *Int. J. Robotics Research*, 10(5):481–491, 1991. first(?) solution for integrable inverse kinematics via definition of joint-space impedance.

[175] R. Nair and J. H. Maddocks. On the forward kinematics of parallel manipulators. *Int. J. Robotics Research*, 13(2):171–188, 1994.

[176] Y. Nakamura. *Advanced robotics: redundancy and optimization.* Addison-Wesley, Reading, MA, 1991.

[177] Y. Nakamura and H. Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *ASMEDSMC*, 108:163–171, 1986.

[178] Y. Nakamura, H. Hanafusa, and T. Yoshikawa. Task-priority based redundancy control of robot manipulators. *Int. J. Robotics Research*, 6(2):3–15, 1987.

[179] P. Nanua. Direct kinematics of parallel mechanisms. Master's thesis, Ohio State University, Ohio, 1988.

[180] 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.

[181] P. Nanua, K. J. Waldron, and V. Murthy. Direct kinematic solution of a Stewart platform. *IEEE Trans. Rob. Automation*, 6(4):438–444, 1990.

[182] J. Naudet. *Forward dynamics of multibody systems: a recursive Hamiltonian approach.* PhD thesis, Vrije Universiteit Brussel, Departement Werktuigkunde, 2005.

[183] J. Naudet, D. Lefeber, and Z. Terze. Forward dynamics of open-loop multibody mechanisms using an efficient recursive algorithm based on canonical momenta. *Multibody Systems Dynamics*, 10(1):45–59, 2003.

[184] J. I. Neimark and N. A. Fufaev. *Dynamics of nonholonomic systems*, volume 33 of *Translations of mathematical monographs.* American Mathematical Society, Providence, RI, 1972.

[185] E. Nelson. *Tensor analysis.* Princeton University Press, Princeton, NJ, 1967.

[186] I. Newton. *Philosophiae naturalis principia mathematica.* Maclehose, Glasgow, Scotland, reprinted edition, 1871. Eds. W. Thomson and H. Blackburn.

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

[188] 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.

[189] 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.

[190] D. E. Orin and W. W. Schrader. Efficient computation of the Jacobian for robot manipulators. *Int. J. Robotics Research*, 3(4):66–75, 1984.

[191] 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, Guimaraẽs, Portugal, 1996.

[192] F. C. Park. Computational aspects of the Product-of-Exponentials formula for robot kinematics. *IEEE Trans. Autom. Control*, 39(3):643–647, 1994.

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

[194] F. C. Park. Optimal robot design and differential geometry. *Trans. ASME J. Mech. Design*, 117:87–92, 1995.

[195] 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.

[196] 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.

[197] S. Patarinski, H. Van Brussel, and H. Thielemans. Kinematics and control of wheeled mobile robots. In *Intelligent Robots*, pages 177–186, Bangalore, India, 1993.

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

[199] R. Penrose. A generalized inverse for matrices. *Proc. Cambridge Philos. Soc.*, 51:406–413, 1955.

[200] J. Peters, M. Mistry, F. Udwadia, J. Nakanishi, and S. Schaal. A unifying framework for robot control with redundant DOFs. *Autonomous Robots*, 24(1):1–12, 2008.

[201] D. L. Pieper. *The kinematics of manipulators under computer control*. PhD thesis, Stanford University, Department of Mechanical Engineering, 1968.

[202] F. Pierrot, A. Fournier, and P. Dauchez. Towards a fully-parallel 6 dof robot for high-speed applications. In *Int. Conf. Robotics and Automation*, pages 1288–1293, Sacramento, CA, 1991.

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

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

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

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

[207] E. P. Popov, A. F. Vereshchagin, and S. L. Zenkevich. *Manipulyatsionnye roboty: dinamika i algoritmy*. Nauka, Moscow, 1978.

[208] M. Raghavan. The Stewart platform of general geometry has 40 configurations. *Trans. ASME J. Mech. Design*, 115:277–282, 1993.

[209] M. Raghavan and B. Roth. Inverse kinematics of the general 6R manipulator and related linkages. *Trans. ASME J. Mech. Design*, 115:502–508, 1993.

[210] 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.

[211] C. Reboulet and T. Berthomieu. Dynamic models of a six degree of freedom parallel manipulator. In *Int. Conf. Advanced Robotics*, Pisa, Italy, 1991.

[212] M. Renaud. Coordinated control of robots-manipulators: determination of the singularities of the Jacobian matrix. In *Proceedings of the First Yugoslav Symposium on Industrial Robotics and Artificial Intelligence*, pages 153–165, 1979.

[213] M. Renaud. Calcul de la matrice Jacobienne nécessaire à la commande coordonnée d'un manipulateur. *Mechanism and Machine Theory*, 15(2):81–91, 1980.

[214] F. Reuleaux. *The kinematics of machinery.* MacMillan, London, England, 1876. Republished 1963 by Dover Publ. Co., New York.

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

[216] R. E. Roberson and R. Schwertassek. *Dynamics of multibody systems.* Springer-Verlag, 1988.

[217] 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.

[218] 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. *J. de Mathématiques Pures et Appliquées*, 5:380–440, 1840.

[219] J. I. Rodríguez, J. M. Jiménez, F. J. Funes, and J. G. de Jalón. Recursive and residual algorithms for the efficient numerical integration of multi-body systems. *Multibody System Dynamics*, 11:295–320, 2004.

[220] 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.

[221] 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.

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

[223] 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.

[224] S. K. Saha and J. Angeles. Kinematics and dynamics of a three-wheeled 2-dof AGV. In *Int. Conf. Robotics and Automation*, pages 1572–1577, Scottsdale, AZ, 1989.

[225] 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.

[226] A. d. Saint Germain. Sur la fonction s introduite par M. Appell dans les équations de la dynamique. *Comptes Rendus de l'Académie des Sciences de Paris*, 130:1174–1177, 1900.

[227] C. Samson and K. Ait-Abderrahim. Feedback control of a nonholonomic wheeled cart in Cartesian space. In *Int. Conf. Robotics and Automation*, pages 1136–1141, Sacramento, CA, 1991.

[228] 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.

[229] 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.

[230] F. A. Scheck. *Mechanics, from Newton's laws to deterministic chaos.* Springer, Berlin, Germany, 2nd edition, 1994.

[231] 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.

[232] L. Sciavicco and B. Siciliano. A solution algorithm to the inverse kinematic problem for redundant manipulators. *IEEE Trans. Rob. Automation*, 4(4):403–410, 1988.

[233] L. Sciavicco and B. Siciliano. *Modeling and Control of Robot Manipulators*. McGraw-Hill, 1996.

[234] T. Shamir. The singularities of redundant robot arms. *Int. J. Robotics Research*, 9(1):113–121, 1990.

[235] T. Shamir and Y. Yomdin. Repeatability of redundant manipulators: Mathematical solution of the problem. *IEEE Trans. Autom. Control*, 33(11):1004–1009, 1988.

[236] D. B. Silver. On the equivalence of Lagrangian and Newton-Euler dynamics for manipulators. *Int. J. Robotics Research*, 1(2):60–70, 1982.

[237] M. Skreiner. A study of the geometry and the kinematics of instantaneous spatial motion. *J. of Mechanisms*, 1:115–143, 1966.

[238] M. Skreiner. On the points of inflection in general spatial motion. *J. of Mechanisms*, 2:429–433, 1967.

[239] 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.

[240] M. W. Spong and M. Vidyasagar. *Robot dynamics and control*. Wiley, New York, NY, 1989.

[241] 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.

[242] 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.

[243] E. Staffetti, H. Bruyninckx, and J. De Schutter. On the invariance of manipulability indices. In J. Lenarčič and F. Thomas, editors, *Advances in Robot Kinematics*, pages 57–66, Caldes de Malavella, Spain, 2002. Kluwer.

[244] M. M. Stanišić and O. Duta. Symmetrically actuated double pointing systems: The basis of singularity-free robot wrists. *IEEE Trans. Rob. Automation*, 6(5):562–569, 1990.

[245] B. Steer. Trajectory planning for a mobile robot. *Int. J. Robotics Research*, 8(5):3–14, 1989.

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

[247] G. Strang. *Introduction to applied mathematics*. Wellesley-Cambridge Press, Wellesley, MA, 1986.

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

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

[250] 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.

[251] K. Sugimoto, J. Duffy, and K. H. Hunt. Special configurations of spatial mechanisms and robot arms. *Mechanism and Machine Theory*, 17:119–132, 1982.

[252] C. H. Suh and C. W. Radcliffe. *Kinematics and mechanisms design*. Wiley, New York, NY, 1978.

[253] A. N. A. N. T. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Mathematics Doklady (Doklady Akademii Nauk SSSR)*, 151(3):501–504, 1963.

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

[255] 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.

[256] Z. S. Tumeh and C. O. Alford. Solving for manipulator joint rates in singular positions. In *Int. Conf. Robotics and Automation*, pages 987–992, Philadelphia, PA, 1988.

[257] 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.

[258] L. Van Aken. *Robot motions in free space: task specification and trajectory planning.* PhD thesis, Dept. Mech. Eng., Katholieke Univ. Leuven, Belgium, 1987.

[259] W. K. Veitschegger and C.-H. Wu. Robot accuracy analysis based on kinematics. *IEEE J. Rob. Automation*, RA-2(3):171–179, 1986.

[260] A. F. Vereshchagin. Computer simulation of the dynamics of complicated mechanisms of robot-manipulators. *Eng. Cybern.*, 12(6):65–70, 1974.

[261] A. F. Vereshchagin. Gauss principle of least constraint for modelling the dynamics of automatic manipulators using a digital computer. *Soviet Physics Doklady*, 20(1):33–34, 1975. Originally published in Dokl. Akad. Nauk SSSR, Vol. 220, No. 1, pp. 51–53, 1975.

[262] A. F. Vereshchagin. Modelling and control of motion of manipulational robots. *Soviet J. of Computer and Systems Sciences*, 27(5):29–38, 1989. Originally published in Izvestiia Akademii nauk SSSR, Tekhnicheskaya Kibernetika, No. 1, pp. 125–134, 1989.

[263] A. F. Vereshchagin and V. L. Generozov. Design of trajectories of operating device of automatic manipulator. *Eng. Cybern.*, 16:55–64, 1979.

[264] A. F. Vereshchagin, V. L. Generozov, and V. B. Krucherov. Algorithm for control of velocity vector manipulator. *Eng. Cybern.*, 13(3):51–55, 1975.

[265] R. von Mises. Motorrechnung, ein neues Hilfsmittel der Mechanik. *Z. Angew. Math. 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.

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

[267] 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.

[268] C. W. Wampler. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares solutions. *IEEE Trans. on Systems, Man, and Cybernetics*, 16:93–101, 1986.

[269] 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.

[270] 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.

[271] H. J. Warnecke and R. D. Schraft. *Industrie-Roboter.* Krausskopf Verlag, 1973.

[272] D. A. Wells. *Lagrangian dynamics.* Schaum's Outline Series. McGraw-Hill, New York, NY, 1967.

[273] P. Wenger. A new general formalism for the kinematic analysis of all nonredundant manipulators. In *Int. Conf. Robotics and Automation*, pages 442–447, Nice, France, 1992.

[274] 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.

[275] D. E. Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Trans. Man-Machine Systems*, 10(2):47–53, 1969.

[276] D. E. Whitney. The mathematics of coordinated control of prosthetic arms and manipulators. *Trans. ASME J. Dyn. Systems Meas. Control*, 94:303–309, 1972.

[277] J. Wiitala, B. J. Rister, and J. P. Schmiedeler. A more flexible robotic wrist. *Mechanical Engineering*, 119(7):78–80, 1997.

[278] 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.

[279] J. Wittenburg. *Dynamics of systems of rigid bodies.* Teubner, Stuttgart, Germany, 1977.

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

[281] W. A. Wolovich. *Robotics: basic analysis and design.* Holt, Rinehart and Winston, New York, NY, 1986.

[282] W. A. Wolovich and H. Elliot. A computational technique for inverse kinematics. In *Proc. 23rd IEEE Conf. Decision and Control*, pages 1359–1363, 1984.

[283] L. S. Woo and F. Freudenstein. Application of line geometry to theoretical kinematics and the kinematic analysis of mechanical systems. *J. of Mechanisms*, 5:417–460, 1970.

[284] G. Wu, S. Siegler, P. Allard, C. Kirtley, A. Leardini, D. Rosenbaum, M. Whittle, D. D. D'Lima, L. Cristofolini, H. Witte, O. Schmid, and I. Stokes. ISB recommendation on definitions of joint coordinate system of various joints for the reporting of human joint motion–Part I: ankle, hip, and spine. *J. Biomechanics*, 35(4):543–548, 2002.

[285] G. Wu, F. C. T. van der Helm, H. E. J. Veeger, M. Makhsous, P. Van Roy, C. Anglin, J. Nagels, A. R. Karduna, K. McQuade, X. Wang, F. W. Werner, and B. Buchholz. ISB recommendation on definitions of joint coordinate systems of various joints for the reporting of human joint motion–Part II: shoulder, elbow, wrist and hand. *J. Biomechanics*, 38(5):981–992, 2005.

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

[287] 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.

[288] T. Yoshikawa. Manipulability of robotic mechanisms. *Int. J. Robotics Research*, 4(2):3–9, 1985.

[289] W. Yourgrau and S. Mandelstam. *Variational principles in dynamics and quantum theory.* Pitman and Sons, London, England, 3rd edition, 1968.

[290] 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.

[291] 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.

[292] 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.

[293] 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.

[294] H. Ziegler. *Mechanics*, volume 2. Addison-Wesley, Reading, MA, 1966.