h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

University of Applied Sciences Darmstadt

– Computer Science Department –

Lecture Handout

Introduction to Robotics

Simulation of Robotic Systems

Prof. Dr. Thomas Horsch

Prof. Dr. Thomas Horsch: *Lecture Handout*, Introduction to Robotics / Simulation of Robotic Systems, © version 0.1

FOREWORD

This lecure handout is primarily addressed to students, who are participating my lectures *Introduction to Robotics* and/or *Simulation of Robotic Systems* at the Univiersity of Applied Sciences Darmstadt, Germany.

The lecture *Simulation of Robotic Systems* is addressed to mechatronic students (major in robotics) and to computer science students (elective lecture) in their bachelor program. Mechatronic students already have a basic robotic backgroud through previous lectures, computer science students usually do not have. In order to have a self contained handout, this handout also includes an introduction to robotics. Computer science students are expected to learn this part through self studying.

During the course of this handout I use external sources, which - I hope - are fully referenced. A very good source is the textbook [**lynch**]. The authors provide at their homepage `http://hades.mech.northwestern.edu/index.php/Modern_Robotics` further resources including short videos and a printable version of their book.

xxxx

The practical exercises of these courses typically contain 5 to 6 tasks. As an underlying software tool we use the educational version of CoppeliaSim `https://www.coppeliarobotics.com/` for programming and simulation.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

AGV	Automatic Guided Vehicle
DOF	Degree of Freedom
IFR	International Federation of Robotics
TCP	Tool Center Point

Part I

LECTURE HANDOUT

1 INRODUCTION

Robotics is an interdisciplinary field that integrates computer science and engineering. Robotics involves design, construction, operation, and use of robots. The goal of robotics is to design machines that can help and assist humans. Robotics integrates fields of mechanical engineering, electrical engineering, information engineering, mechatronics, electronics, bioengineering, computer engineering, control engineering, software engineering, among others.

Robotics develops machines that can substitute for humans and replicate human actions. Robots can be used in many situations and for many purposes, but today many are used in dangerous environments (including inspection of radioactive materials, bomb detection and deactivation), manufacturing processes, or where humans cannot survive (e.g. in space, underwater, in high heat, and clean up and containment of hazardous materials and radiation). Robots can take on any form but some are made to resemble humans in appearance. This is said to help in the acceptance of a robot in certain replicative behaviors usually performed by people. Such robots attempt to replicate walking, lifting, speech, cognition, or any other human activity. Many of today's robots are inspired by nature, contributing to the field of bio-inspired robotics.

Certain robots require user input to operate while other robots function autonomously. The concept of creating robots that can operate autonomously dates back to classical times, but research into the functionality and potential uses of robots did not grow substantially until the 20th century. Throughout history, it has been frequently assumed by various scholars, inventors, engineers, and technicians that robots will one day be able to mimic human behavior and manage tasks in a human-like fashion. Today, robotics is a rapidly growing field, as technological advances continue; researching, designing, and building new robots serve various practical purposes, whether domestically, commercially, or militarily. Many robots are built to do jobs that are hazardous to people, such as defusing bombs, finding survivors in unstable ruins, and exploring mines and shipwrecks. Robotics is also used in STEM (science, technology, engineering, and mathematics) as a teaching aid.

Industrial robots are mainly used for

- Automation of industrial tasks
- Spot welding
- Arc welding
- Machine transfer and machine tending

- Pick and Place
- Clueing and sealing
- Deburring
- Assembly/Disassembly
- Testbed for Artificial Intelligence (AI)

Robotics is an interdisciplinary subject requiring expertise in the following fields

- Computer Science
- Mechanical Engineering
- Electrical and Electronic Engineering
- Mathematics

It is worth noting that many problems in computer graphics animation are related to robotics problems (much of the material in this course is relevant to animation).

1.1 MOTIVATION

As the field of robotics grows ever more sophisticated, a greater number of engineers are required to lend their talents to design, program, and maintain robots and robotic systems. Not surprisingly, the complexity of these machines and systems has spawned five specialized areas within the field of robotics:

- Operator interface
- Mobility or locomotion
- Manipulators and Effectors
- Programming
- Sensing and Perception

Since the development of today's most advanced robotic systems is no easy endeavor, those tasked with their design, programming and maintenance often look to hone in on a particular field of expertise. These sections will explore those fields in greater detail. [**onlinerobotics**]

Operator Interface

A robot is only as good as its ability to effectively communicate with a human controller. The operator interface – commonly referred to as a Human Robot Interface – is the medium that allows the user and the robot to communicate. Most specifically, it is the method by which a human operator can give pre-programmed commands for the robot to execute. A gaming controller is an example of a basic **hri!** (**hri!**). It allows a player to issue a set of commands to the system, which are then executed in the game. In manufacturing, an industrial touchscreen computer on a piece of equipment or in a centralized control room is also a form of **hri!**. The operator can issue commands to the conveyor or other device to execute on the factory floor.

A great deal of care needs to go into the design of HRIs. They must be intuitive to use, and enable operators to communicate effectively with the robot, in order to execute tasks accurately and efficiently.

Mobility or Locomotion

In order for a robot to complete a task, it needs to be able to move in its environment. In robotics, this movement is called locomotion. Mobility in robotics is achieved in many different ways. For example, some robots mimic human movement, like those used on assembly lines or those whose design is based on human anatomy. Flying robots and drones make use of propellers and other propulsion systems. Other robots, such as the rovers deployed on Mars and other celestial bodies, require wheels to get around. In short, the environment a robot will be used in often determines how the engineer will design the mobility system.

Manipulators and Effectors

For any robot to be worthwhile, it must be able to interact with its environment; that's where manipulators and effectors come into play. These are the parts of the robot that allow it to pick up objects and move them, or manipulate items that are separate from the system. Human-like robots will employ appendages and digits that work like human hands, in order to complete a given task. In industrial settings, manipulators and effectors are perhaps more commonly represented by pincers, claws, or pushers which are all uniquely suited to move heavy pieces of equipment or materials. Like the other disciplines listed in this article, having the foundational knowledge received from robotics technician training can prepare aspiring robotics engineers for specializing in this area of robotics.

Programming

Programming is essentially the language an operator uses to communicate with the robot. Traditionally, any action that a robot was required to perform had to be programmed. These days, advanced programming allows robotic systems to learn and adapt to changes within its environment.

Generally speaking, commands can be provided by the user in real time for the robot to perform, or the robot can be programmed to perform a series of tasks, in sequence, autonomously. Regardless of the method the commands are given, each robot can be programmed using one of many different programming languages, so an engineer looking to specialize in this particular field of robotics will have a lot to become proficient in.

Sensing and Perception

Robots use sensors to gather information. This information lets the robot know the physical space it occupies, where it needs to go, and if any obstacles block its path. Sensors also collect information to help the robot decide how to react to objects it encounters. The right sensor must be selected for each robot's specific application to ensure that the correct decisions are made.

As the field of robotics expands with integration across industries, so too will the demand for experienced robotics engineers to maintain these technologies.

1.2 TYPES OF ROBOTS

- Mobile robots
- Industrial robots
- Service robot
- Educational robot
- Modular robot
- Collaborative robot

Mobile robot

Mobile robots have the capability to move around in their environment and are not fixed to one physical location. An example of a mobile robot that is in common use today is the automated guided vehicle or automatic guided vehicle Automatic Guided Vehicle (AGV). An AGV is a mobile robot that follows markers or wires in the floor, or uses vision or lasers.

Mobile robots are also found in industry, military and security environments. They also appear as consumer products, for entertainment or to perform certain tasks like vacuum cleaning. Mobile robots are the focus of a great deal of current research and almost every major university has one or more labs that focus on mobile robot research.

Mobile robots are usually used in tightly controlled environments such as on assembly lines because they have difficulty responding to unexpected interference. Because of this most humans rarely encounter robots. However domestic robots for cleaning and maintenance are increasingly common in and around homes in developed countries. Robots can also be found in military applications.

Industrial robot (manipulating)

Industrial robots usually consist of a jointed arm (multi-linked manipulator) and an end effector, that is attached to a fixed surface. One of the most common type of end effector is a gripper assembly.

The International Organization for Standardization gives a definition of a manipulating industrial robot in ISO 8373: *"an automatically controlled, reprogrammable, multipurpose, manipulator programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation applications."*

This definition is used by the International Federation of Robotics (IFR) and many national standards committees.

Service robot

Most commonly industrial robots are fixed robotic arms and manipulators used primarily for production and distribution of goods. The term "service robot" is less well-defined. The International Federation of Robotics has proposed a tentative definition, *"A service robot is a robot which operates semi- or fully autonomously to perform services useful to the well-being of humans and equipment, excluding manufacturing operations."*

Educational (interactive) robot

Robots are used as educational assistants to teachers. From the 1980s, robots such as turtles were used in schools and programmed using the Logo language. There are robot kits like Lego Mindstorms, BIOLOID, OLLO from ROBOTIS, or BotBrain Educational Robots can help children to learn about mathematics, physics, programming, and electronics. Robotics have also been introduced into the lives of elementary and high school students in the form of robot competitions with the company FIRST (For Inspiration and Recognition of Science and Technology). The organization is the foundation for the FIRST Robotics Competition, FIRST LEGO League, Junior FIRST LEGO League, and FIRST Tech Challenge competitions.

Modular robot

Modular robots are a new breed of robots that are designed to increase the utilization of robots by modularizing their architecture. The functionality and effectiveness of a modular robot is easier to increase compared to conventional robots. These robots are composed of a single type of identical, several different identical module types, or similarly shaped modules, which vary in size. Their architectural structure allows hyper-redundancy for modular robots, as they can be designed with more than 8 DOF. Creating the programming, inverse kinematics and dynamics for modular robots is more complex than with traditional robots. Modular robots may be composed of L-shaped modules, cubic modules, and U and H-shaped modules. ANAT technology, an early modular robotic technology patented by Robotics Design Inc., allows the creation of modular robots from U and H shaped modules that connect in a chain, and are used to form heterogeneous and homogenous modular robot systems. These "ANAT robots" can be designed with "n" DOF as each module is a complete motorized robotic system that folds relatively to the modules connected before and after it in its chain, and therefore a single module allows one degree of freedom. The more modules that are connected to one another, the more degrees of freedom it will have. L-shaped modules can also be designed in a chain, and must become increasingly smaller as the size of the chain increases, as payloads attached to the end of the chain place a greater strain on modules that are further from the base. ANAT H-shaped modules do not suffer from this problem, as their design allows a modular robot to distribute pressure and impacts evenly amongst other attached modules, and therefore payload-carrying capacity does not decrease as the length of the arm increases. Modular robots can be manually or self-reconfigured to form a different robot, that may perform different applications. Because modular robots of the same architecture type are composed of modules that compose different modular robots, a snake-arm robot can combine with another to form a dual or quadra-arm robot, or can split into several mobile robots, and mobile robots can split into multiple smaller ones, or combine with others into a larger or different one. This allows a single modular robot the ability to be fully specialized in a single task, as well as the capacity to be specialized to perform multiple different tasks.

Modular robotic technology is currently being applied in hybrid transportation, industrial automation, duct cleaning and handling. Many research centres and universities have also studied this technology, and have developed prototypes.

Collaborative robot

A collaborative robot or cobot is a robot that can safely and effectively interact with human workers, while performing simple industrial tasks. However, end-effectors and other environmental conditions may create hazards, and as such risk assessments should be done before using any industrial motion-control application. The collaborative robots most widely used in industries today are manufactured by Universal Robots in Denmark.

Rethink Robotics—founded by Rodney Brooks, previously with iRobot - introduced Baxter in September 2012; as an industrial robot designed to safely interact with neighboring human workers, and be programmable for performing simple tasks. Baxters stop if they detect a human in the way of their robotic arms and have prominent off switches.

1.3 OBJECTIVES

....

1.4 STRUCTURE

....

2 FOUNDATIONS

In this chapter the basic terminology and mathematical foundations are presented

2.1 TERMINOLOGY

We begin with some robot terminology for **industrial robots**.

Exactly what constitutes a robot is sometimes debated. Numerically controlled machines are usually not referred as robots. The distiction lies somewhere in the sophistication of the programmability of the device – if a mechanical device can be programmed to perfom a wide variety of applications, it is probably an industrial robot. Machines which are for the most part relegated to one class of task are considered fixed automation.

Industrial robots consist of (nearly) rigid links which are connected with joints which allow relative motion of neighboring links. These joints are usually instrumented with position sensors which allow the relative position of neighboring links to be measured. In case of rotational or revolute joints, these displacements are called joint angles. Some robots contain sliding, or prismatic joints.

The number of degrees of freedom that a robot possesses is the number of independant position varaibles, which would have to be specified in order to locate all parts of the mechanism. Usually a robot is an open kinematic chain. This implies, that each joint variable is usually defined with a single variable and the number of joints equals the number of degrees of freedom.

At the free end of the chain of links which make up the robot is the end effector. Depending on the intended application of the robot, the end effector may be a gripper, welding torch or any other device. We generally describe the position of the manipulator by giving a description of the tool frame or sometimes called the Tool Center Point (TCP), which is attached to the end-effector, relative to the base frame, which is attached to the nonmoving base of the manipulator.

Local coordinate systems are used to describe the position and orientation of the rigid bodies in space. They serve as a reference system within which to express the position and orientation of this body.

Exercises with CoppeliaSim: Load sample scene files (.ttt) and play around.

Robots from different robot vendors are available: ABB, KUKA, Universal Robots and others.

Kinematics is the science of motion which treats motion without regard to the forces which cause it. Within the science of kinematics one studies the position, velocity, acceleration, and all higher order derivatives of the position variables. Hence, the study of the kinematics of robots refers to all geometrical and time-based properties of the motion.

A very basic problem to be solved is:

How to relate the robot's configuration or pose to the position and orientation of its end effector. A configuration of an n-degree of freedom robot is an n-vector $(q_1, ..., q_n)$, where each q_i is typically either a rotational joint angle or a prismatic joint translation. This is known as the forward kinematics of the robot. This is the static geometrical problem of computing the position and orientation of the end-effector of the robot. Specifically, given a set of joint angles, the forward kinematic problem is to compute the position and orientation of the TCP relative to the base frame. Sometimes we think of this as changing the representation of robot position from joint space description into a cartesian space description.

The following problem is considered the inverse kinematics: Given the position and orientation of the end-effector of the robot, calculate all possible sets of joint angles which could be used to attain the this given position and orientation. The inverse kinematics is not as simple as the forward kinematics. Because the kinematic equations are nonlinear, their solution is not always easy or even possible in a closed form. The existence of a kinematic solution defines the workspace of a given robot. The lack of a solution means that the robot cannot attain the desired position and orientation because it lies outside the robot's workspace.

In addition to dealing with static positioning problems, we may wish to analyse robots in motion. Often in performing velocity analysis of a mechanism it is convenient to define a matrix quantity called the jacobian of the robot. The jacobian specifies a mapping from velocities in joint space to velocities in cartesian space. The nature of this mapping changes as the configuration of the robot varies. At certain points, called singularities, this mapping is not invertible. An understanding of this phenomenon is important to designers and users of robots.

A common way of causing a robot to move from A to B in a smooth, controlled fashion is to cause each joint to move as specified by a smooth function of time. Commonly, each joint starts and ends its motion at the same time, so that the robot motion appears coordinated. Exactly how to compute these motion functions is the problem of trajectory generation.

A robot programming language serves as an interface between the human user and the industrial robot. Central questions arise such as:

- How are motions through space described easily by a programmer?
- How are multiple robots programmed so that they can work in parallel?
- How are sensor-based actions described in a language?

The sophistication of the user interface is becoming extremely important as robots and other programmable devices are applied to more and more demanding industrial applications.

An off-line programming system is a robot programming environment which has been sufficiently extended, generally by means of computer graphics, so that the development of robot programs can take place without access to the robot itself. A common argument raised in the favor is that an off-line programming system will not cause production equipment (i.e. the robot) to be tied up when it needs to be reprogrammed; hence, automated factories can stay in production mode a greater percentage of time.

An industrial robot constists of three main systems:

PROGRAMMING SYSTEM: A robot user needs to „teach" the robot the specific task which is to be performed. This can be done with the so called teach in method. The programmer uses the teach box of the robot control and drives the robot to the diesired positions and stores them and other values like travel speed, corner smoothing parameters, process parameters, etc. For the programming period which can take a significant time for complex tasks the production is idle. Another possibility is the use of Off-line programming systems mentioned before.

ROBOT CONTROL: The robot control interprets the robot's application program and generates a series of joint values and joint velocities (and sometimes accellerations) in an appropriate way for the feedback control. Today mainly an independant joint control approach is taken, but more and more sophisticated control approaches, like adaptive control, nonlinear control, etc are involved in robotics. Figure 2.1 shows a possible control structure.

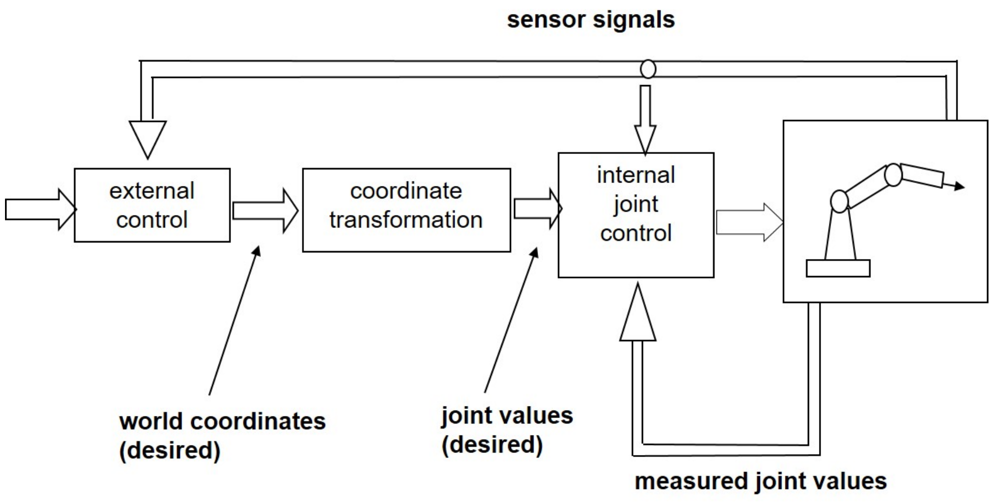

Figure 2.1: This block diagram shows the structure of a simple robot control (internal and external)

ROBOT MECHANICS: The robot mechanics will transform the joint torques applied by the servo drives into an appropriate motion. Usually AC motors are used to drive the axis. Resolvers serve as a position sensor and are normally located on the shaft of the actuator. Resolvers are devices which output two analog signals – one is the sine of the shaft angle and the other the cosine. The shaft angle is determined from the relative magnitude of the two signals.

2.2 CONFIGURATION SPACE

The content of this section is taken from [**lynch:2019**].
A robot is mechanically constructed by connecting a set of bodies, called links, to each other using various types of joints. Actuators, such as electric motors, deliver forces or torques that cause the robot's links to move. Usually an endeffector, such as a gripper or hand for grasping and manipulating objects, is attached to a specific link. All the robots considered in this document have links that can be modeled as rigid bodies.
Perhaps the most fundamental question one can ask about a robot is, where is it?
The answer is given by the robot's configuration: a specification of the positions of all points of the robot.

For example, the configuration of a door can be represented by a single number, the angle θ about its hinge. The configuration of a point on a plane can be described by two coordinates, (x, y). The configuration of a coin lying heads up on a flat table can be described by three coordinates: two coordinates (x, y) that specify the location of a particular point on the coin, and one coordinate θ, that specifies the coin's orientation. The above coordinates all take values over a continuous range of real numbers.
The number of degrees of freedom (DOF) of a robot is the smallest number of real-valued coordinates needed to represent its configuration.

In the example in figure 2.2, the door has one degree of freedom and the point (in2D) two degrees of freedom. The coin lying heads up on a table has three degrees of freedom.

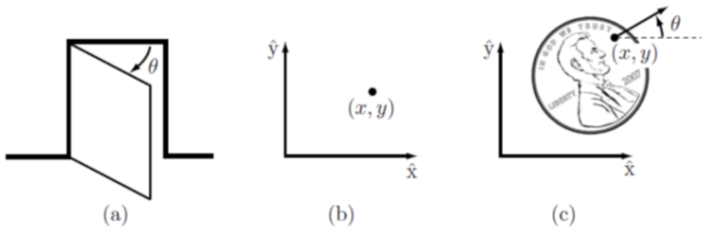

Figure 2.2: Examples of configurations of a door, point in 2D and a coin

2.3 SPATIAL TRANSFORMATIONS

The following section summarizes the main mathematical methods in order to describe positions and orientations with respect to a particular coordinate system. Typically a robotic application is described in a 3D world. In such a 3D world we assume each object to be a rigid body. For each 3D-body a reference coordinate system (sometimes also called a reference frame) is defined. This system is attached to a specific point on or in the object. Typically (like CoppeliaSim does) the center of mass is used to attach the reference frame to an object, but any other point can do as well. We will start with some notations. The content of this section is mainly taken from [**lynch:2019**].

2.3.1 *Positions, Orientations and Coordinate Systems*

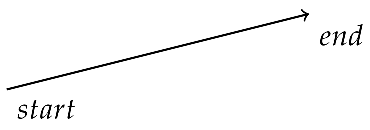

Figure 2.3: A vector pointing from *start* to *end*.

A free vector is a geometric quantity with a length and a direction. Think of it as an arrow in $\mathbb{R}^n$. It is called "free" because it is not necessarily rooted anywhere; only its length and direction matter. A free vector can be translated without change (two vectors are identical, if they equal in length and direction).

In the context of a reference frame (coordinate system) a (three-dimensional) vector is a three-tupel of numerical values indicating the distances along the axes of the coordinate system. Each of these distances along an axis can be thought of as a result of projecting the vector onto the corresponding axis.

A point P in physical space can also be represented as a vector. Given a choice of reference frame and length scale for physical space, the point P can be represented as a vector from the reference frame origin to P; its vector representation is denoted in italics by $p \in \mathbb{R}^n$. Here, as before, a different choice of reference frame and length scale for physical space leads to a different representation $p \in \mathbb{R}^n$ for the same point P in physical space, see figure 2.4: If we fix a reference frame $\{a\}$, with unit coordinate axes $\hat{x}_a$ and $\hat{y}_a$, we can represent p as $p_a = (1, 2)$. If we fix a reference frame $\{b\}$ at a different location, a different orientation, and a different length scale, we can represent p as $p_b = (4, -2)$.
In the rest of this document, a choice of length scale will always be assumed, but we will be dealing with reference frames at different positions and orientations. A reference frame can be placed anywhere in space, and any reference frame leads to an equally valid representation of the underlying space and the objects in it. We always assume that exactly one stationary fixed frame, or space frame, denoted $\{s\}$, has been defined. This might be attached to a corner of a room, for example. In robotics this system is sometimes also

called the *world coordinate system*. Similarly, we often assume that at least one frame has been attached to some moving rigid body, such as the body of a quadcopter flying in the room. This body frame, denoted $\{b\}$, is the stationary frame that is coincident with the body-attached frame at any instant. While it is common to attach the origin of the $\{b\}$ frame to some important point on the body, such as its center of mass, this is not necessary. The origin of the $\{b\}$ frame does not even need to be on the physical body itself, as long as its configuration relative to the body, viewed from an observer stationary relative to the body, is constant.

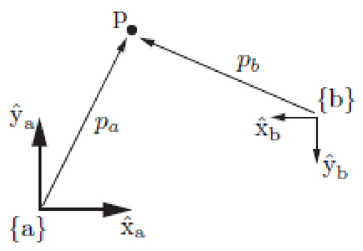

Figure 2.4: The point p exists in physical space, and it does not care how we represent it.

Important! All frames in this document are stationary frames. When we refer to a body frame $\{b\}$, we mean a motionless frame that is instantaneously coincident with a frame that is fixed to a (possibly moving) body.
For simplicity, we will usually refer to a body frame as a frame attached to a moving rigid body. Despite this, at any instant, by "body frame" we actually mean the stationary frame that is instantaneously coincident with the frame moving along with the body. All reference frames are right-handed, as illustrated in figure 2.5. For right-handed reference frames the three-finger-rule of the right hand is used: Thumb points in (positive) x-direction, pointing finger in (positive) y-direction, middle finger point then in (positive) z-direction.

A positive rotation about an axis is defined as the direction in which the fingers of the right hand curl when the thumb is pointed along the axis (figure 2.6).

In figure 2.7 two 3D reference frames are shown. The red axis denotes the x-axis, the green axis the y-axis and the blue axis the z-axis. The left system is the located in the origin of the world's coordinate system. The right system is positioned relative to left system by a displacement in x and y by 0.3 meters and a rotation around the z-axis by $45° \widehat{=} \frac{\pi}{4}$.

In order to present 3D reference frames on a 2D plane like this sheet of paper, such systems could be projected in the 2D plane. Then one axis of the coordinate systems points inside or outside the drawing plane. To distiguish both cases, an axis, which points outside the drawing plane is shown as $\odot$, while an axis pointing inside is shown as $\otimes$. The figure 2.7 drawn projected in 2D is shown in figure 2.8.

Figure 2.5: Three finger rule illustrated on the Swiss 200-francs banknote [**rhs**]

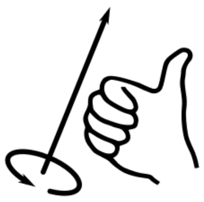

Figure 2.6: Right hand rule for rotation [**chegg**]

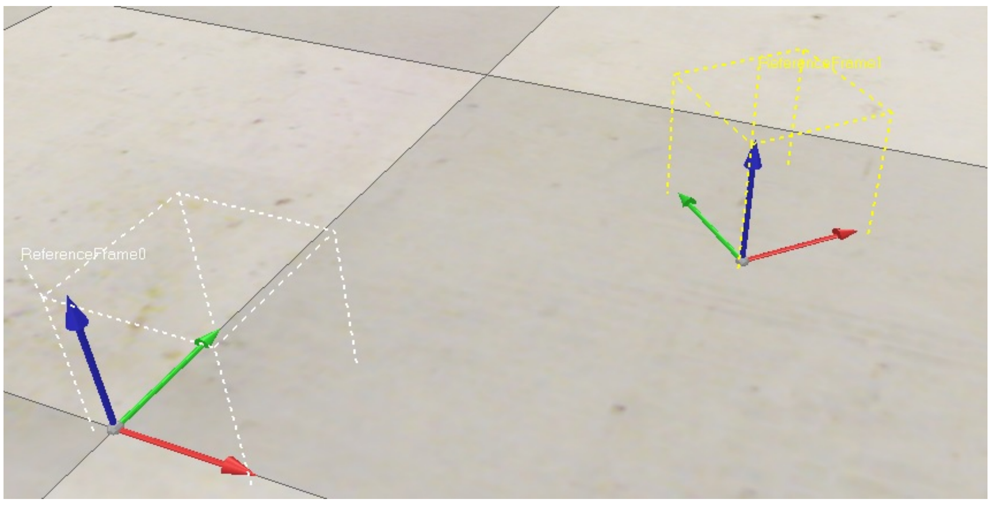

Figure 2.7: Two 3D reference systems in a CoppeliaSim scence

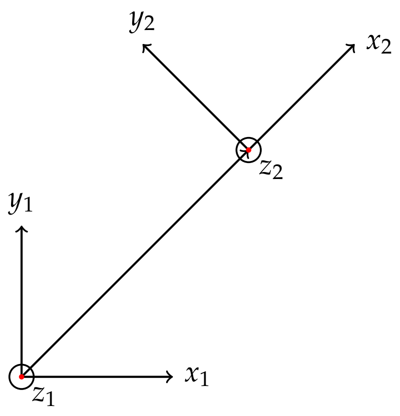

Figure 2.8: Two reference systems projected in 2D, both z-axes point out of the drawing plane

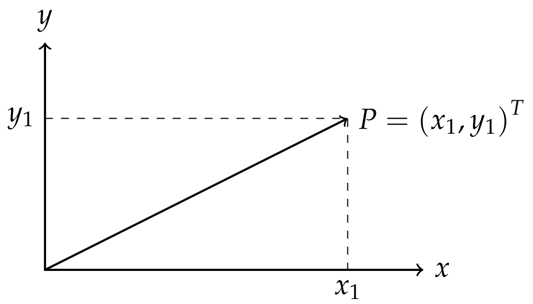

Figure 2.9: A 2D-vector in a coordinate system

Once a reference frame is established we can locate any point in the world with 3x1 position vector. Sometimes more than one coordinate system are defined. So vectors must be tagged with information identifying which coordinate system they are defined within. In this document vectors are written with a leading subscript indicating the coordinate system to which they are referenced (unless it is clear from context, this subscript is omitted). Geometrically a vector can be interpreted as an arrow describing its length and direction.

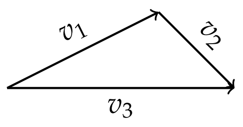

Figure 2.10: Addition of two vectors in 2D: $v_3 = v_1 + v_2$.

$$v_3 = v_1 + v_2 = \begin{pmatrix} v_{1x} + v_{2x} \\ v_{1y} + v_{2y} \\ v_{1z} + v_{2z} \end{pmatrix}$$

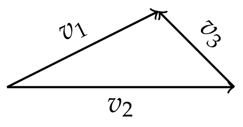

Figure 2.11: Substraction of two vectors in 2D: $v_3 = v_1 - v_2$.

$$v_3 = v_1 - v_2 = \begin{pmatrix} v_{1x} - v_{2x} \\ v_{1y} - v_{2y} \\ v_{1z} - v_{2z} \end{pmatrix}$$

The right-hand rule tells us that the thumb points in the direction of angular velocity when we hold the axis with the right-hand and curl the fingers in the direction of motion of the rotating body. Any body rotating in a fixed axis can rotate in a clockwise or counter-clockwise direction when viewed along the axis. When the object is rotating clockwise, the direction of angular velocity is along the axis of the circular path directed downwards. When the object is rotating counter-clockwise, the direction of angular velocity is along with the circular path directed upwards.

2.3.2 *Rotations*

Rotations can be described in different ways. We will discuss here rotation matrix, angle-axis representation, euler angles and quaternions.

2.3.2.1 *Rotation matrix*

A rotation matrx R is a 3 by 3 matrix with special propertiers. R has nine entries, only three can be chosen independently, since a rotation has 3 DOF.

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}.$$

We begin by expressing a set of six explicit constraints on the entries of R. The three columns of R correspond to the reference frame unit axes $\hat{x}, \hat{y}$ and $\hat{z}$. The following conditions must therefore be satisfied:

The unit norm condition $\hat{x}$, $\hat{y}$ and $\hat{z}$ are all unit vectors:

$$r_{11}^2 + r_{21}^2 + r_{31}^2 = 1$$
$$r_{12}^2 + r_{22}^2 + r_{32}^2 = 1$$
$$r_{13}^2 + r_{23}^2 + r_{33}^2 = 1$$

The orthogonality condition $\hat{x}^T\hat{y} = \hat{x}^T\hat{z} = \hat{y}^T\hat{z} = 0$:

$$r_{11}r_{12} + r_{21}r_{22} + r_{31}r_{32} = 0$$
$$r_{11}r_{13} + r_{21}r_{23} + r_{31}r_{33} = 0$$
$$r_{12}r_{13} + r_{22}r_{23} + r_{32}r_{33} = 0$$

These six constraints can be expressed more compactly as a single set of constraints on the matrix R:

$$R^T * R = R * R^T = I$$

where R^T denotes the transpose of R and I denotes the identity matrix. So inverting a rotation matrix is easy, you just need to transpose it $R = (r_{ij})$ then $R^T = (r_{ji})$.

Properties of a rotation matrix:

- $\det(R) = 1$
- columns are mutually orthogonal
- possible to describe rotations with 3 parameters
- same result by investigation of R: six constraints and nine matrix elements
- rotations do not generally commute: $R_1 * R_2 \neq R_2 * R_1$

Exercise: Calculate $R_X(\frac{\pi}{2})R_Z(\frac{\pi}{2})$ and $R_Z(\frac{\pi}{2})R_X(\frac{\pi}{2})$ and solve both compound transformations geometrically by drawing a frame diagram.

2.3.2.2 *Angle-axis representation*

The axis–angle representation is equivalent to the more concise rotation vector, also called the Euler vector. In this case, both the rotation axis and the angle are represented by a vector codirectional with the rotation axis whose length is the rotation angle θ:

$$\boldsymbol{\theta} = \theta \mathbf{e}$$

$\mathbf{e}$ has unit length.

2.3.2.3 *Quaternions*

The quaternion number system extends the complex numbers. Quaternions were first described by Irish mathematician William Rowan Hamilton in 1843. Unit quaternions provide a convenient mathematical notation for representing spatial orientations and rotations of elements in three dimensional space. Specifically, they encode information about an axis-angle rotation about an arbitrary axis.

Unit quaternions describe rotations.

$$q = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} = \begin{pmatrix} \cos(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2})e_x \\ \sin(\frac{\theta}{2})e_y \\ \sin(\frac{\theta}{2})e_z \end{pmatrix}$$

Exercise: Proof, that q has unit length.

The decription of a unit quaterion as a rotation is not unique. The negative of a unit represents the same rotation

$$q = \begin{pmatrix} \cos(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2})e_x \\ \sin(\frac{\theta}{2})e_y \\ \sin(\frac{\theta}{2})e_z \end{pmatrix} = \begin{pmatrix} \cos(\frac{\theta+2\pi}{2}) \\ \sin(\frac{\theta+2\pi}{2})e_x \\ \sin(\frac{\theta+2\pi}{2})e_y \\ \sin(\frac{\theta+2\pi}{2})e_z \end{pmatrix} = \begin{pmatrix} \cos(\frac{\theta}{2}+\pi) \\ \sin(\frac{\theta}{2}+\pi)e_x \\ \sin(\frac{\theta}{2}+\pi)e_y \\ \sin(\frac{\theta}{2}+pi)e_z \end{pmatrix} = \begin{pmatrix} -\cos(\frac{\theta}{2}) \\ -\sin(\frac{\theta}{2})e_x \\ -\sin(\frac{\theta}{2})e_y \\ -\sin(\frac{\theta}{2})e_z \end{pmatrix} = -q$$

This property is important for interpolating rotations. If you use unit quaternions to interpolate rotations you first check, whether the second quaternion lies in the same 4D halfsphere or not:

$$\arccos(q_1^T q_2) > \pi$$

In that case q_2 (or q_1) is negated to lie in the same halfspere.

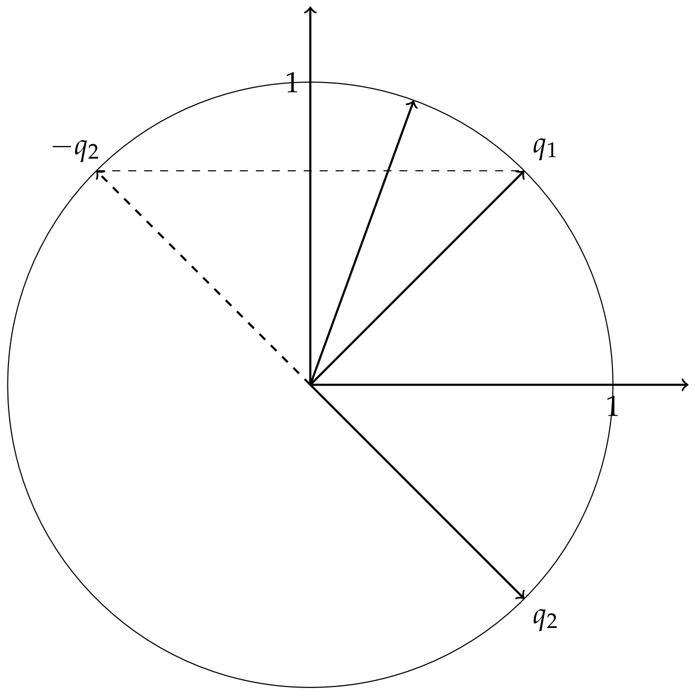

Figure 2.12: Quaternion interpolation from q_1 to q_2

2.3.3 *Euler angles: Yaw, pitch, and roll*

Euler angles are three angles introduced by Leonhard Euler to describe the orientation of a rigid body with respect to a fixed coordinate system. The angles describe three successive rotations of one reference system to align it with the other one. Since the order of rotation is important, you must specify it.

One possiblity is to use so called yaw, pitch and roll angles.

A 3D body can be rotated about three orthogonal axes, as shown in figure 2.16. Borrowing aviation terminology, these rotations will be referred to as yaw, pitch, and roll:

1. A yaw is a counterclockwise rotation of α about the z-axis. The rotation matrix is given by

$$R_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.1}$$

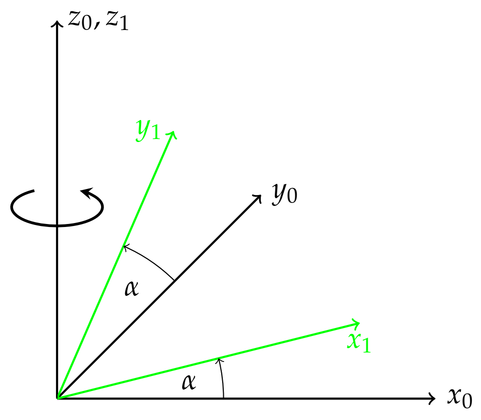

Figure 2.13: Angle α rotates around the z-axis

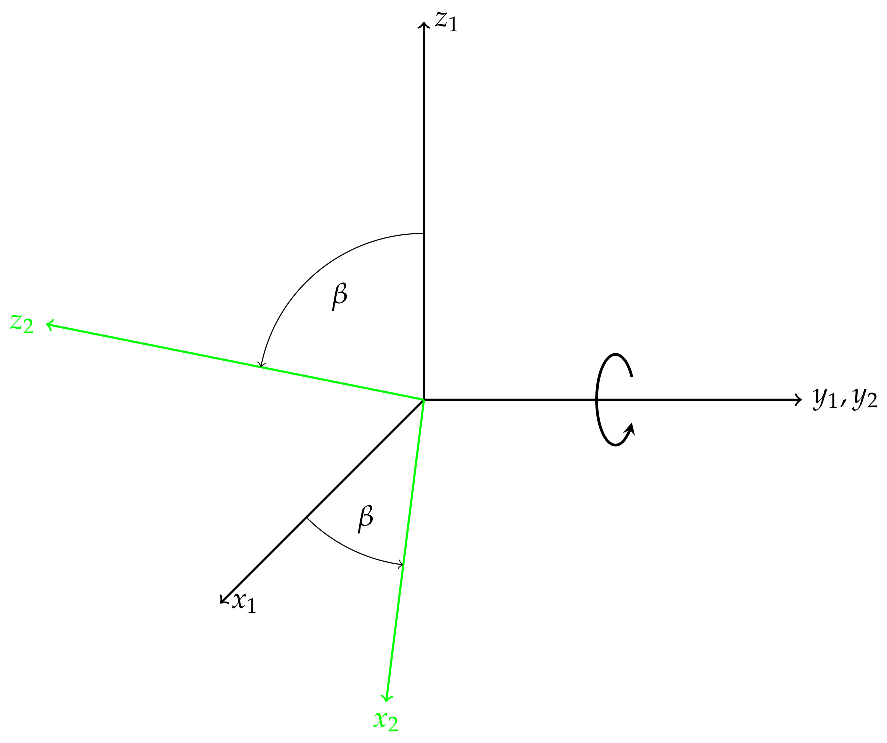

Figure 2.14: Angle β rotates around the y-axis

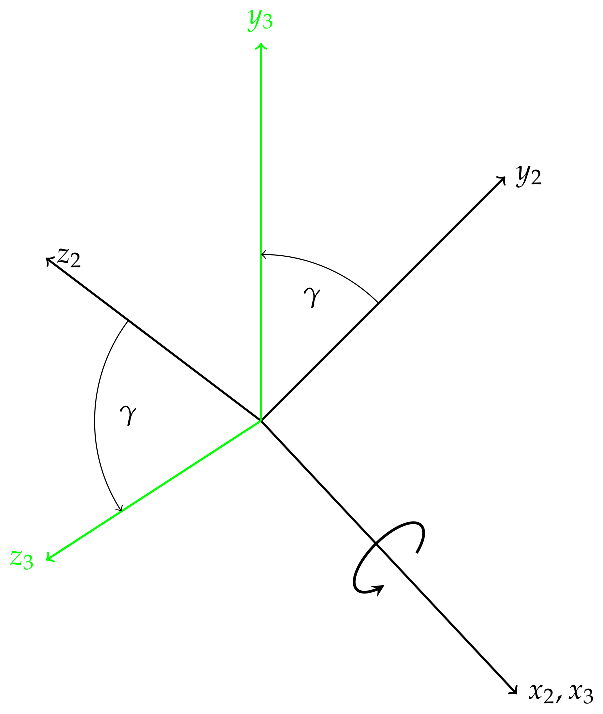

Figure 2.15: Angle γ rotates around the x-axis

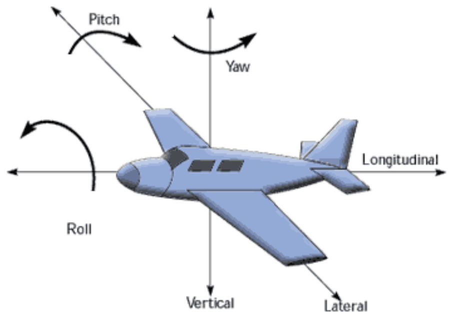

Figure 2.16: Roll, pitch and yaw of an airplane [**novatel**]

Note that the upper left entries of $R_z(\alpha)$ form a 2D rotation applied to the x and y coordinates, whereas the z coordinate remains constant.

2. A pitch is a counterclockwise rotation of β about the y-axis. The r otation matrix is given by

$$R_y(\beta) = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix}. \tag{2.2}$$

3. A roll is a counterclockwise rotation of γ about the x-axis. The rotation matrix is given by

$$R_x(\gamma) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{pmatrix}. \tag{2.3}$$

The yaw, pitch, and roll rotations can be used to place a 3D body in any orientation. A single rotation matrix can be formed by multiplying the yaw, pitch, and roll rotation matrices to obtain

$$R(\alpha,\beta,\gamma) = R_z(\alpha)\,R_y(\beta)\,R_x(\gamma) = \begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{pmatrix}. \tag{2.4}$$

It is important to note that $R(\alpha,\beta,\gamma)$ performs the roll first, then the pitch, and finally the yaw. If the order of these operations is changed, a different rotation matrix would result. Be careful when interpreting the rotations. Consider the final rotation, a yaw by α. Imagine sitting inside of a robot $\mathcal{A}$ that looks like an aircraft. If $\beta = \gamma = 0$, then the yaw turns the plane in a way that feels like turning a car to the left. However, for arbitrary values of β and γ, the final rotation axis will not be vertically aligned with the aircraft because the aircraft is left in an unusual orientation before α is applied. The yaw rotation occurs about the z-axis of the world frame, not the body frame of $\mathcal{A}$. Each time a new rotation matrix is introduced from the left, it has no concern for original body frame of $\mathcal{A}$. It simply rotates every point in $\mathbb{R}^3$ in terms of the world frame. Note that 3D rotations depend on three parameters, α, β, and γ, whereas 2D rotations depend only on a single parameter, θ. The primitives of the model can be transformed using $R(\alpha,\beta,\gamma)$, resulting in $\mathcal{A}(\alpha,\beta,\gamma)$.

Determining yaw, pitch, and roll from a rotation matrix

It is often convenient to determine the α, β, and γ parameters directly from a given rotation matrix. Suppose an arbitrary rotation matrix

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \tag{2.5}$$

is given. By setting each entry equal to its corresponding entry in 2.4, equations are obtained that must be solved for α, β, and γ. Note that $r_{21}/r_{11} = \tan\alpha$ and $r_{32}/r_{33} = \tan\gamma$. Also, $r_{31} = -\sin\beta$ and $\sqrt{r_{32}^2 + r_{33}^2} = \cos\beta$.
If $r_{31} \neq \pm 1$, solving for each angle yields:

$$\alpha = \tan^{-1}(r_{21}/r_{11})$$

$$\beta = \tan^{-1}\left(-r_{31}/\sqrt{r_{32}^2 + r_{33}^2}\right)$$

and

$$\gamma = \tan^{-1}(r_{32}/r_{33})$$

If $r_{31} = -1$, then $\beta = \frac{\pi}{2}$

$$R(\alpha, \frac{\pi}{2}, \gamma) = R_z(\alpha)\, R_y(\frac{\pi}{2})\, R_x(\gamma) = \begin{pmatrix} 0 & \cos\alpha\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\cos\gamma + \sin\alpha\sin\gamma \\ 0 & \sin\alpha\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\cos\gamma - \cos\alpha\sin\gamma \\ -1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -\sin(\alpha-\gamma) & \cos(\alpha-\gamma) \\ 0 & \cos(\alpha-\gamma) & \sin(\alpha-\gamma) \\ -1 & 0 & 0 \end{pmatrix}$$

A one-parameter family of solutions for α and γ exists. One possible solution is $\alpha = 0$ and $\gamma = \tan^{-1}(r_{12}/r_{22})$.

If $r_{31} = 1$, then $\beta = -\frac{\pi}{2}$, and a one-parameter family of solutions for α and γ exists. One possible solution is $\alpha = 0$ and $\gamma = -\tan^{-1}(r_{12}/r_{22})$.

There is a choice of four quadrants for the inverse tangent functions. How can the correct quadrant be determined? Each quadrant should be chosen by using the signs of the numerator and denominator of the argument. The numerator sign selects whether the direction will be above or below the x-axis, and the denominator selects whether the direction will be to the left or right of the y-axis. This is the same as the atan2 function in the C programming language, which nicely expands the range of the arctangent to $[0, 2\pi)$. This can be applied to the above equations as

$$\alpha = \text{atan2}(r_{21}, r_{11}), \beta = \text{atan2}\left(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}\right), \gamma = \text{atan2}(r_{32}, r_{33})$$

Note that this method assumes $r_{11} \neq 0$ and $r_{33} \neq 0$.

A second possiblity is to reverse the order:

$$R(\alpha, \beta, \gamma) = R_x(\alpha)\, R_y(\beta)\, R_z(\gamma)$$

A third possiblity is to take another order using the z-axis twice:

$$R(\alpha, \beta, \gamma) = R_z(\alpha)\, R_y(\beta)\, R_z(\gamma)$$

Euler angles are easy to interpret, since the always rotate consecutively about the axes x, y or z, but they do not have as good interpolation properties compared to unit quaternions. One issue is the so called gimbal lock. Gimbal lock is the loss of one degree of freedom in a three-dimensional, three-gimbal mechanism that occurs when the axes of two of the three gimbals are driven into a parallel configuration, "locking" the system into rotation in a degenerate two-dimensional space.

The word lock is misleading: no gimbal is restrained. All three gimbals can still rotate freely about their respective axes of suspension. Nevertheless, because of the parallel orientation of two of the gimbals' axes there is no gimbal available to accommodate rotation about one axis.
This Link shows a nice simulation of some properties of euler angles.
This Link shows a nice simulation of the gimbal lock problem using euler angles.
Exercise: Find out, which kind of representation is used in CoppeliaSim.

2.3.4 *Homogenous transformations*

We now consider representations for the combined orientation and position of a rigid body. A natural choice would be to use a rotation matrix R to represent the orientation of the body frame b in the fixed frame s and a vector $p \in \mathbb{R}^3$ to represent the origin of b in s. Rather than identifying R and p separately, we package them into a single 4 by 4 matrix as follows. This matrix is called a homogenous transformation

In other fields of study it can be used to compute perspective and scaling operations. In our case they can be used to describe the spatial relationship between two reference systems.
The main advantage using homogenous transforms is to express consecutive transforms as a matrix multiplication.

$$T = \left[\begin{array}{ccc|c} & R & & \mathbf{p} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{c|c} R & \mathbf{p} \\ \hline \mathbf{0}^T & 1 \end{array}\right] = \begin{bmatrix} R & \mathbf{p} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

This 4 by 4 matrix describes a tranformation of a reference system to another. Since it consists of a rotation and a translation, the order is relevant. A pure translation with no rotation described as a transformation is:

$$Trans = \begin{bmatrix} I & p \\ 0^T & 1 \end{bmatrix}$$

A pure rotation with no translation described as a transformation is:

$$Rot = \begin{bmatrix} R & 0 \\ 0^T & 1 \end{bmatrix}$$

A transformation as a product of a pure rotation and a pure translation:

$$Rot * Trans = \begin{bmatrix} R & 0 \\ 0^T & 1 \end{bmatrix} * \begin{bmatrix} I & p \\ 0^T & 1 \end{bmatrix} = \begin{bmatrix} R & Rp \\ 0^T & 1 \end{bmatrix}$$

A transformation as a product of a pure rtranslation and a pure rotation:

$$Trans * Rot = \begin{bmatrix} I & p \\ 0^T & 1 \end{bmatrix} * \begin{bmatrix} R & 0 \\ 0^T & 1 \end{bmatrix} = \begin{bmatrix} R & p \\ 0^T & 1 \end{bmatrix}$$

Exercise: Invert the transformation T and write the inverse in block format.

There are three major uses for a transformation matrix T:

1. to represent the configuration (position and orientation) of a rigid body;
2. to change the reference frame in which a vector or frame is represented;
3. to displace a vector or frame.

3 KINEMATICS

In this chapter we discuss the topic kinematics with the following properties:

- A robot may be thought of as a set of bodies connected in a chain by joints (open serial chain of links).
- These bodies are called links.
- Normally robots consist of joints with one degree of freedom (1 **DOF! (DOF!)**).
- n-DOF joints can be modeled as n joints with 1 DOF connected with n-1 links of zero length.
- Positioning a robot in 3-space a minimum of six joints is required, since the end-effector needs 6 DOF for a general task.
- Typical industrial robot consist of 6 joints.
- Robot with more than 6 DOF in 3D space is called a redundant robot, it has more flexibility (dexterity)
- Joint axis are defined by lines in space.
- Two consecutive links can have a "crooked position".
- Link length: measured along the line which is mutually perpendicular to both axes.
- Link twist: measured in the plane defined by the perpendicular axis.

3.1 FORWARD KINEMATICS

The forward kinematics of a robot refers to the calculation of the position and orientation of its end-effector frame from its joint coordinates θ. Figure 3.2 illustrates the forward kinematics problem for a 3R planar open chain. The link lengths are L_1, L_2 and L_3. Choose a fixed frame 0 with origin located at the base joint as shown, and assume an end-effector frame 4 has been attached to the tip of the third link. The Cartesian position (x, y) and orientation θ of the end-effector frame as functions of the joint angles $(\theta_1, \theta_2, \theta_3)$ are then given by

If one is only interested in the (x, y) position of the end-effector, the robot's task space is then taken to be the x–y-plane, and the forward kinematics would consist of the first two equations only. If the end-effector's position and orientation both matter, the forward kinematics would consist of the three equations.

For a simple robot the forward kinematics can be calculated by basic trigonometry. This is shown in figure **??**

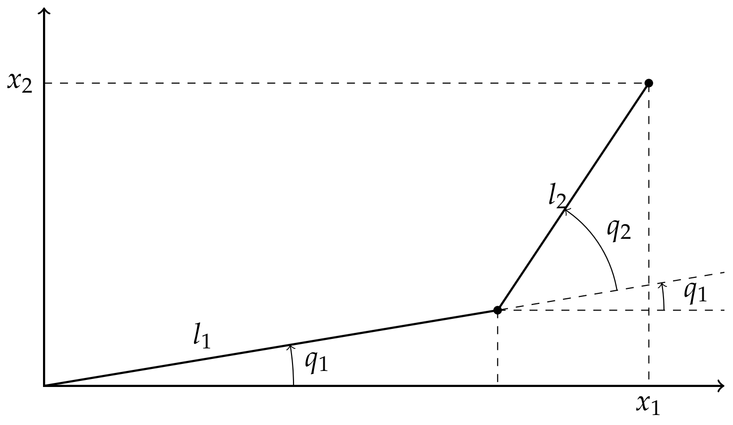

Figure 3.1: Forward kinematics of a 2R planar open chain.

The coordinates of the end-effector (TCP) can be calculated by basic trigonometry:

$$\mathbf{f}(\mathbf{q}) = \mathbf{f}(q_1, q_2) = \begin{pmatrix} f_1(q_1, q_2) \\ f_2(q_1, q_2) \end{pmatrix} = \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) \\ l_1 \sin(q_1) + l_2 \sin(q_1 + q_2) \end{pmatrix}$$

While the above analysis can be done using only basic trigonometry, it is not difficult to imagine that for more general spatial chains the analysis can become considerably more complicated. A more systematic method of deriving the forward kinematics might involve attaching reference frames to each link; in figure **??** the three link reference frames are respectively labeled $1, 2$ and 3.

The chain of tranformations from the robot base system 0 to the end-effector system 4 can be described as follows:

$$T_{04} = \underbrace{I * R_Z(\theta_1)}_{T_{01}} * \underbrace{Tr_X(L_1) * R_Z(\theta_2)}_{T_{12}} * \underbrace{Tr_X(L_2) * R_Z(\theta_3)}_{T_{23}} * \underbrace{Tr_X(L_3)}_{T_{34}}$$

The forward kinematics can then be written as a product of four homogeneous transformation matrices:

$$T_{04} = T_{01} * T_{12} * T_{23} * T_{34}$$

with T_{ij} describing the transformation from reference system i to reference system j. Each transformation T_{ij} itself consitsts of the constant transformation to the next joint and the variable part, rotating around an axis or translating along an axis.

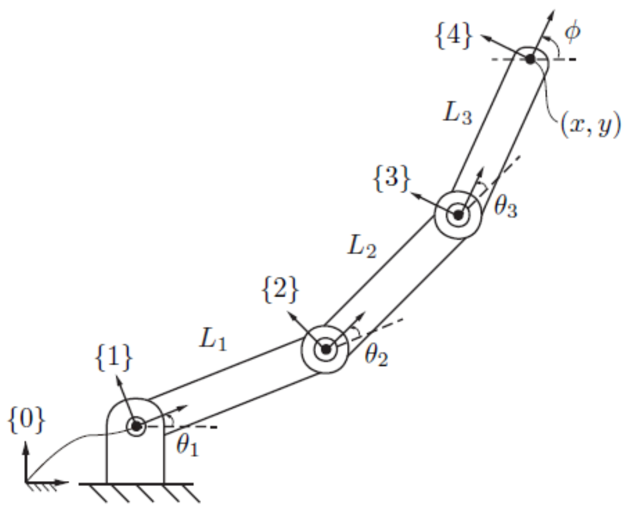

Figure 3.2: Forward kinematics of a 3R planar open chain. For each frame, the $\hat{x}$- and $\hat{y}$-axis is shown; the $\hat{z}$-axis is pointing out of the page.

$$T_{01} = \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} T_{12} = \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & L_1 \\ \sin\theta_2 & \cos\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_{23} = \begin{pmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & L_2 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} T_{34} = \begin{pmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Observe that T_{34} is constant and that each remaining $T_{i-1,i}$ depends only on the joint variable θ_i. As an alternative to this approach, let us define M to be the position and orientation of frame $\{4\}$ when all joint angles are set to zero (the "home" or "zero" position of the robot). Then

$$T_{04} = \begin{pmatrix} 1 & 0 & 0 & L_1 + L_2 + L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

In general you can define the kinematics of a serially linked robot in form of a table. For that, you assume the robot beeing in its "zero" (or "home") position. The 3-DOF planar robot of figure **??** could be described in the following way:

Joint	DOF	Tr_x	Tr_y	Tr_z	R_x	R_y	R_z	Remark
	-	0	0	0	0	0	0	$\{0\} \rightarrow \{1\}$
1	$R_z(\theta_1)$	L_1	0	0	0	0	0	$\{1\} \rightarrow \{2\}$
2	$R_z(\theta_2)$	L_2	0	0	0	0	0	$\{2\} \rightarrow \{3\}$
3	$R_z(\theta_3)$	L_3	0	0	0	0	0	$\{3\} \rightarrow \{4\}$

The first row of table 3.1 describes the fact, that the base coordinate system of robot is equal to world coordinate system (WCS). If the robot base coordinate system is different to the WCS, then this row would contain values different from zero.
In general, the first task is to derive a sceleton model of the robot and attach local reference system to each joint. The next step is to identify the axis of rotation of the reference system in case of a rotational joint and the axis along translated in case of a prismatic joint. Having located the reference frames of each link, the transformation from one link to the next must be identified

Let us exercise this with a realistic industrial robot, see figure 3.5:

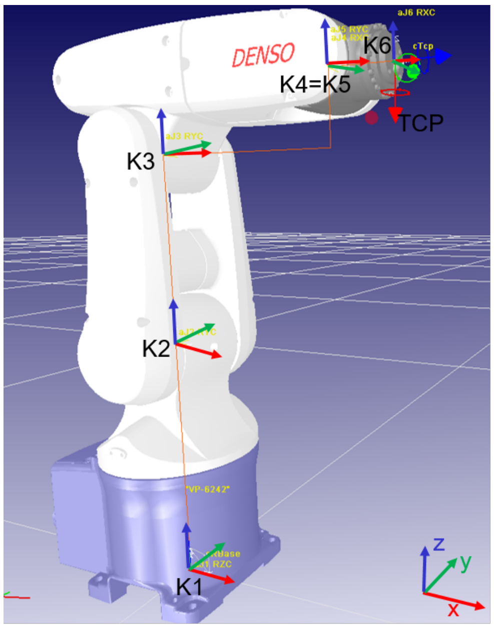

Figure 3.3: A Denso robot with 6 rotational joints in its zero configuration.

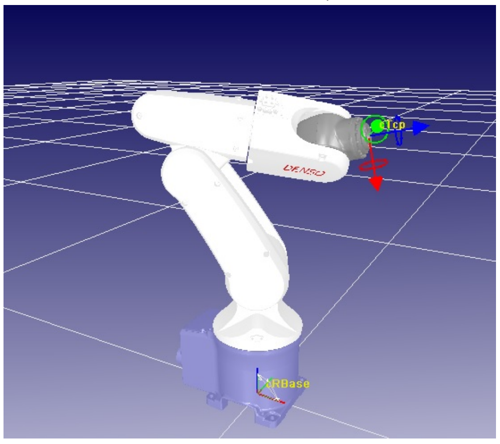

Figure 3.4: A Denso robot in a non zero configuration.

Table 3.1: Kinematics parameter of a Denso robot in its zero configuration in mm and degree.

Joint	DOF	Tr_x	Tr_y	Tr_z	R_x	R_y	R_z	Remark
	-	0	0	0	0	0	0	$WCS \to K_1$
1	$R_z(\theta_1)$	0	0	280	0	0	0	$K_1 \to K_2$
2	$R_y(\theta_2)$	0	0	210	0	0	0	$K_2 \to K_3$
3	$R_y(\theta_3)$	210	0	75	0	0	0	$K_3 \to K_4$
4	$R_x(\theta_4)$	0	0	0	0	0	0	$K_4 \to K_5$
5	$R_y(\theta_5)$	70	0	0	0	0	0	$K_5 \to K_6$
6	$R_x(\theta_6)$	0	0	0	0	90	0	$K_6 \to TCP$

Table 3.2 shows the kinematics parameter of a Denso robot in its zero configuration.

Let us exercise this with a second industrial robot UR5 from Universal Robots, see figure 3.8. Figure 3.9 shows the UR5 robot in its zero configuration projected in 2D, figure 3.10 in 3D.

Exercise: Fill in the kinematic parameters of the UR5 into the following table:

Table 3.2: Kinematics parameter of a Denso robot in its zero configuration in mm and degree.

Joint	DOF	Tr_x	Tr_y	Tr_z	R_x	R_y	R_z	Remark
	-	0	0	0	0	0	0	$WCS \to K_1$
1								$K_1 \to K_2$
2								$K_2 \to K_3$
3								$K_3 \to K_4$
4								$K_4 \to K_5$
5								$K_5 \to K_6$
6								$K_6 \to TCP$

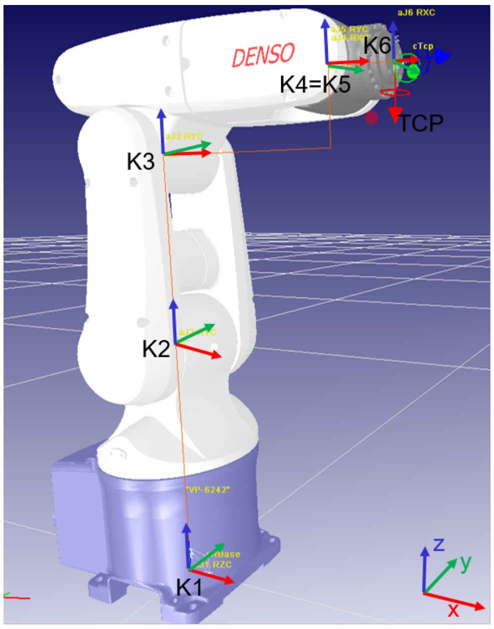

Figure 3.5: A Denso robot with 6 rotational joints in its zero configuration.

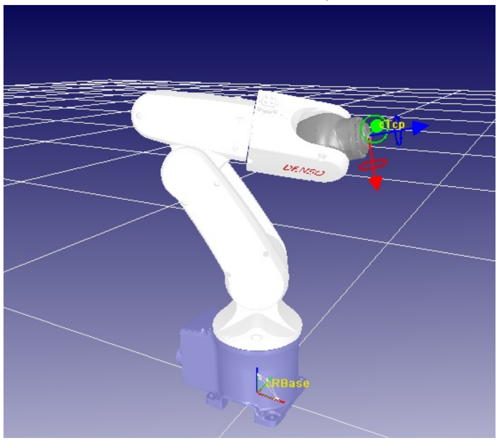

Figure 3.6: A Denso robot in a non zero configuration.

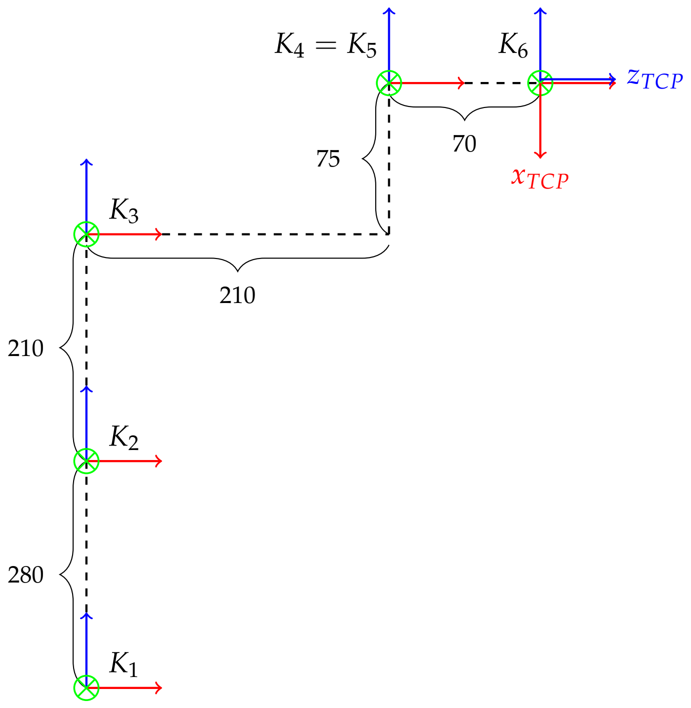

Figure 3.7: Sceleton model of a 6 DOF Denso Robot

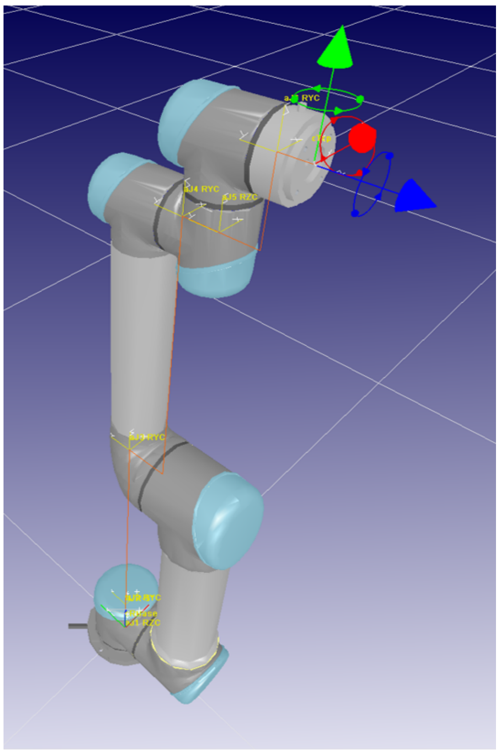

Figure 3.8: UR5 robot with 6 rotational joints in its zero configuration.

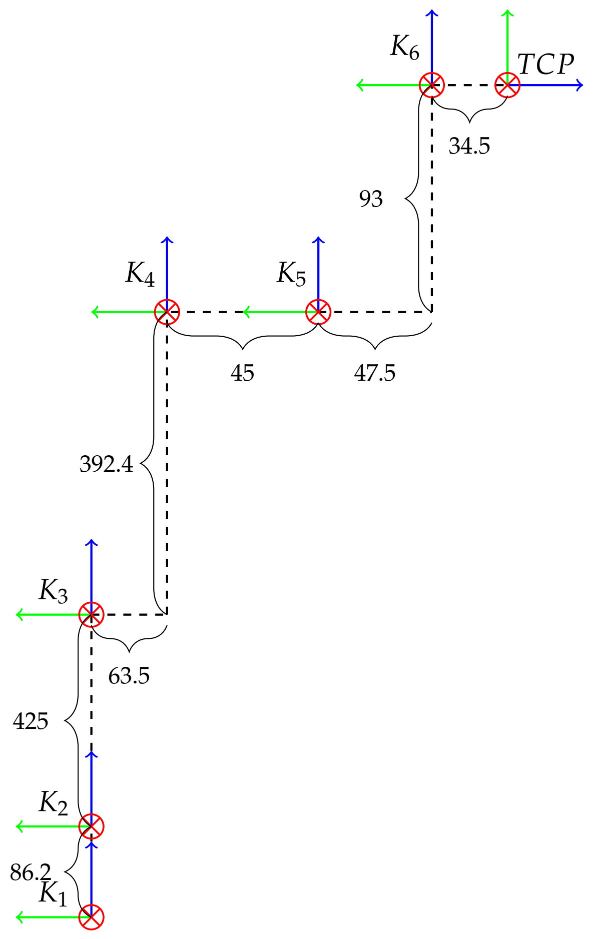

Figure 3.9: Sceleton of UR5 model in 2D (y-z plane)

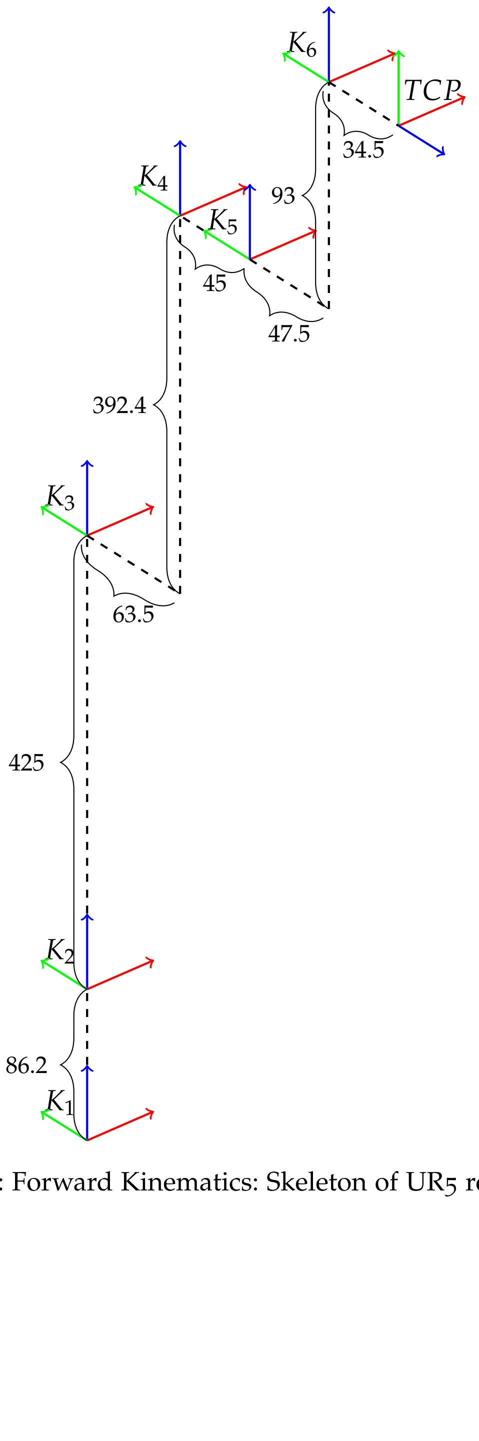

Figure 3.10: Forward Kinematics: Skeleton of UR5 robot in 3D

3.2 INVERSE KINEMATICS

The inverse kinematics of a robot refers to the calculation of the joint coordinates θ from the position and orientation of its end-effector frame. In general the inverse kinemtics problem for serial linked robots is much harder to solve than the calculation of the forward kinematics. Problems related to inverse kinematics are:

- Existence of solutions
- Multiple solutions
- Method of solution

3.2.1 *Existence of solutions*

The existence of solutions is closely linked to the workspace of the robot which is the volume of space which the TCP of the robot can reach. For the solution to exist, the specified goal point must be in the workspace.

- Dexterous workspace: volume of space which the robot's TCP can reach with all orientations
- Reachable workspace: volume of space which the robot's TCP can reach with at least one orientation

Examples:
Two DOF rotational robot with $l_1 = l_2$

- Dexterous workspace: the origin
- Reachable workspace: disc of radius $l_1 + l_2$

Dexterous workspace: the origin Reachable workspace: disc of radius
Two DOF rotational robot with $l_1 \neq l_2$

- Dexterous workspace: empty
- Reachable workspace: ring of outer radius $l_1 + l_2$ and inner radius $|l_1 - l_2|$

3.2.2 *Multiple solutions*

In contrast to the forward kinematics problem, which yields a unique solution of the end-effector frame for a given set of joint angles, the inverse kinematics solution does not generally yield a unique solution for the joint angles in general. This can be already observed with the simple 2 DOF rotational robot. For a given position of the TCP in x, y there are two posible solutions called "elbow up" and "elbow down", see figure 3.11 If you consider the

3 DOF rotational robot, there are infinte solutions for the joint angles $\theta_1, \theta_2, \theta_3$ given the position x, y of the TCP

Since the 6 parameters are necessary to describe an arbitrary position and orientation of the TCP, the degree of freedom of the robot must be at least 6, in order a dexterous workspace exists at all. The robot will be redundant, if it has more than 6 axes in 3D space. Typically an industrial robot consist of 6 links connection 6 rotational joints.

If the last 3 axes intersect in one common intersection point at the wrist, orientation and position of this point is decoupled. The position coordinates of this point will only depend on thew first three axes $\theta_1, \theta_2, \theta_3$. In such a case there could be up to 8 joint configurations leading to the same TCP position and orientation.

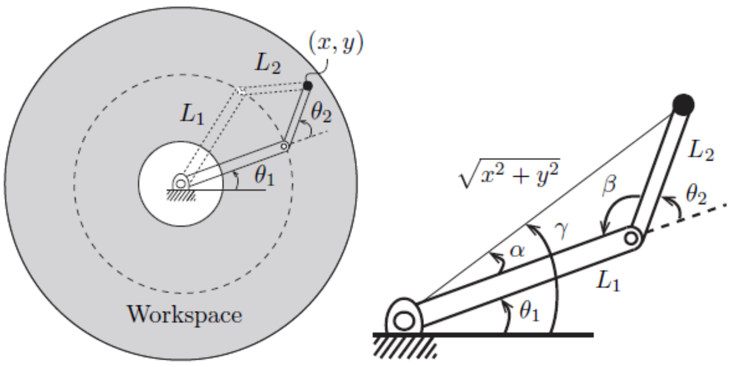

Figure 3.11: Workspace of 2 DOF robot with elbow up and elbow down solution.

3.2.3 *Method of solution*

Formally, given the transformation of the TCP $T(q_1, ..., q_6)$ as 16 numerical values (four of which are trivial), find the joint configurations $(q1, .., q6)_i$, which will result in $T(q_1, ..., q_6)$. Among the 9 equations arising from the rotation matrix portion of T, only three equations are independant. These added with the 3 equations from the position vector portion of T give 6 equations with 6 unknowns. These equations are nonlinear, transcendental equations which can be quite difficult to solve.

In gneral there are two types of methods There are two classes:

- Closed form solutions (analytical, geometrical)
- Numerical solutions

3.2.3.1 *Closed form solution*

For the example in figure 3.1 the angles θ_1 and θ_2 depending on the position x, y of the TCP can to be calculated in a closed form based on the law of cosine.

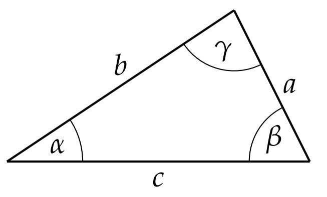

Figure 3.12: law of cosine: $a^2 = b^2 + c^2 - 2bc\cos\alpha$

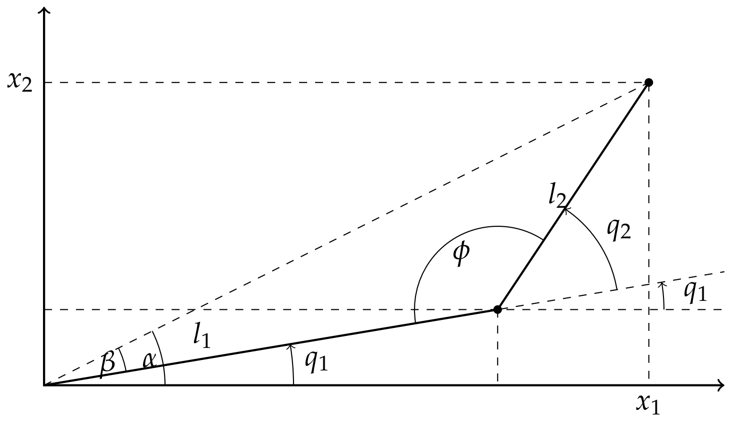

Figure 3.13: Inverse kinematics for a 2 DOF rotational robot

The "elbow down" solution can be formulated:

$$q_1 = \alpha - \beta, \quad \alpha = \arctan(\frac{x_2}{x_1}), \quad \beta = \arccos(\frac{l_1^2 + x_1^2 + x_2^2 - l_2^2}{2l_1\sqrt{x_1^2 + x_2^2}})$$

$$\phi = \arccos(\frac{l_1^2 + l_2^2 - (x_1^2 + x_2^2)}{2l_1 l_2}), \quad q_2 = \pi - \phi$$

Exercise: Calculate the "elbow up" solution.

It depends on the kinematic structure, whether an analytical solution is possible. In case of a 6 DOF robot, where the last 3 axes intersect in one common intersection point at the wrist a closed form can be calclulated.

Part II

APPENDIX