DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Robotics, Cognition, Intelligence

Robust Visual Servo Control and Tracking for the Manipulator of a Planetary Exploration Rover

Laura Maria Bielenberg

DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Robotics, Cognition, Intelligence

Robust Visual Servo Control and Tracking for the Manipulator of a Planetary Exploration Rover

Robustes Visual Servoing und Tracking für den Manipulator eines Rovers zur planetaren Exploration

Author:	Laura Maria Bielenberg
Supervisor:	PD Dr. habil. Rudolph Triebel
Advisors:	Lukas Meyer, M.Sc. and Kristin Bussmann, M.Sc.
Submission Date:	15. June 2021

I confirm that this master's thesis in robotics, cognition, intelligence is my own work and I have documented all sources and material used.

Munich, 05.06.2021 Laura Maria Bielenberg

[illegible]

Acknowledgments

This thesis has been written within the Institute for Robotics and Mechatronics (RM) of the German Aerospace Center (DLR) in Oberpfaffenhofen. The RM accompanied my Master studies over the past two years and I want to thank the entire institute for this exciting, inspiring and insightful time.

I would like to express my special gratitude towards my thesis supervisors Lukas Meyer and Kristin Bussmann, who helped whenever they could and were always interested in my progress. I further thank Lukas Meyer for his time spent proofreading this thesis and his helpful and detailed feedback.

The years of my studies would not have been half as much fun without great friends. I therefore want to thank Alona Sakhnenko, Martin Wudenka, Arne Sachtler, Anna Maria Geissinger, Matthias Bonarens, Anna-Sophie Spath and all of MINT for a wonderful time and tons of beautiful memories.

Special thanks go to Mia, Freddy and Phoebe, the best flatmates anyone could wish for. Lastly, I would like to thank Max Schmachtenberg for his constant support and daily coffee supply, which kept me alive and kicking over the last couple of months.

Abstract

To collect samples and handle tools, planetary exploration rovers commonly employ light-weight robotic manipulators. These can suffer from undesirable positioning imprecision due to erroneous end-effector pose estimates obtained by the manipulators kinematics, leading to the failure of the manipulation task.

This thesis presents a vision-based end-effector pose correction pipeline to improve the positioning precision of the end-effector during manipulation tasks. Our approach corrects the end-effector pose by fusing the estimates obtained by the manipulator's kinematics with information obtained from monocular vision data.

We propose a gradient based method to track a set of active markers within the image stream, which provides us with additional information on the covariance of the retrieved image points. In order to recover the 3D pose estimate of the end-effector, we make use of the maximum likelihood perspective-n-point algorithm, allowing us to propagate the image point uncertainties to their 3D pose covariances. Based on evaluations using recorded ground-truth data, we show that our tracking method leads to a reduction of the kinematic position error by up to 77%.

To operate outdoors and under changing illumination conditions, the robustness of the tracking approach is paramount. Based on the propagated covariance information, we employ an Error-State Kalman filter for the rejection of pose outliers and the reduction of pose jitter. Its smoothing capabilities are confirmed in simulation. We further show the application of the vision-based correction pipeline as part of a visual servoing scheme designed for the collection of payload boxes by the manipulator of the Lightweight Rover Unit 2, developed at the Institute for Robotics and Mechatronics of the German Aerospace Center.

We propose a switching control scheme that applies a position-based visual servo (PBVS) for movements of the end-effector in free space and switches to a PBVS based hybrid impedance visual servoing scheme for movements in close proximity or in direct contact with the coupling partner, to ensure the safe interaction between the manipulator and the payload.

The implemented PBVS approach is lastly evaluated in simulation. We show its successful execution and stability in the vicinity of singularities as well as the avoidance of joint position and velocity limits.

Kurzfassung

Für das Sammeln von Proben und die Handhabung von Werkzeugen nutzen Planetare Explorationsrover in der Regel Manipulatoren in Leichtbauweise. Diese können aufgrund fehlerhafter Schätzungen der auf Grundlage der KinematiK des Manipulators berechneten Endeffektorpose unter Positionierungsungenauigkeiten leiden.

Die vorliegende Arbeit präsentiert eine Computervision basierte Pipeline für die Korrektur der Endeffektorpose der Lightweight Rover Unit 2, zur Nutzung für das Einsammeln von Probencontainern. Der vorgeschlagene Ansatz korrigiert die Ungenauigkeit in der Endeffektorpose durch fusionierung der von Informationen aus Kinematik und monokularen Kameradaten.

Wir stellen eine gradientenbasierte Methode für das visuelle Tracking eines aktiven Markerarrays vor, welche uns zusätzlich Informationen zur Kovarianz der punkte in der Bildebene liefert. Durch die anschließende Nutzung des maximum likelihood perspective-n-point Algorithmus zur Berechnung der 3D-Endeffektorpose, nutzen wir diese Informationen zur Ermittlung der Kovarianz der 3D-Pose. Basierend auf Ground-Truth-Daten zeigen wir, dass die vorgeschlagene Trackingmethode zu einer Reduzierung des kinematischen Positionsfehlers um bis zu 77% führt.

Für den Betrieb im Freien und unter wechselnden Beleuchtungsbedingungen ist die Robustheit des Tracking-Ansatzes von größter Bedeutung. Unter Verwendung der Informationen über die covarianz der Bildprojektionen verwenden wir einen Error-State Kalman Filter zur Unterdrückung von Ausreißern und zur Reduzierung des Rauschens in den ermittelten Endeffektorposen, und bewerten dessen Leistung auf Grundlage simulierter Daten.

Im Anschluss nutzen wir die korrigierte Endeffektorpose als Ausgangspunkt für den Entwurf einer Visual Servoing Strategie zur Präzisionssteigerung während des Andockprozesses zwischen Endeffektor und Probencontainern. Während der Bewegungen des Endeffektors im freien Raum wird ein Positions-basierter Visual Servoing (PBVS) genutzt. Für Bewegungen in unmittelbarer Nähe und bei Kontakt mit dem passiven Kopplungspartner wird ein PBVS-basierter Hybridimpedanzregelungsansatz vorgestellt.

Der implementierte PBVS-Ansatz wird schließlich in Simulation getestet. Wir zeigen dessen erfolgreiche Ausführung und Stabilität in der Nähe von Singularitäten sowie die erfolgreiche Vermeidung von Gelenkpositions- und Geschwindigkeitsbegrenzungen.

Contents

1. Introduction

Space has always been a source of fascination and wonder for humankind. While its exploration has been limited to its passive observation for millennia, the technological advances of the last century made it possible to expand space research to active exploration and probing of planetary surfaces and other celestial bodies.

The use of planetary exploration devices, such as mobile vehicles, also referred to as rovers, allows the investigation of regions so far inaccessible to humans. By supplying images of the planetary terrain and collecting rock and soil samples to be analyzed by scientist back on Earth, such rovers play an important role in the acquisition of new insights that are essential to broaden our understanding about the universe around us.

For use in planetary exploration scenarios, a rover needs to fulfill several requirements. To begin with, it needs to exhibit a lightweight structure to ensure economic transportation to the extraterrestrial operation site. Further, as long distances to potential exploration sites on other celestial bodies such as moon or Mars introduce considerable communication delays, it is desirable that such a rover holds a high degree of autonomy, to ensure its efficient operation [1].

The Lightweight Rover Unit 2 (LRU2), developed at the Institute for Robotics and Mechatronics of the German Aerospace Center (DLR) is a terrestrial prototype of a lightweight rover designed for autonomous planetary exploration, see Fig. 1.1. It is equipped with a six degree of freedom robotic arm, which is used for the collection of soil and rock samples as well as the handling and transportation of payload boxes [1].

1.1. Motivation & Problem Description

One of the main tasks of the LRU2 is the docking, manipulation and transportation of payload boxes, containing soil samples or scientific instruments. Therefore, the correct localization of the target box as well as the accurate pose (position and orientation) estimation of the manipulator's end-effector are paramount.

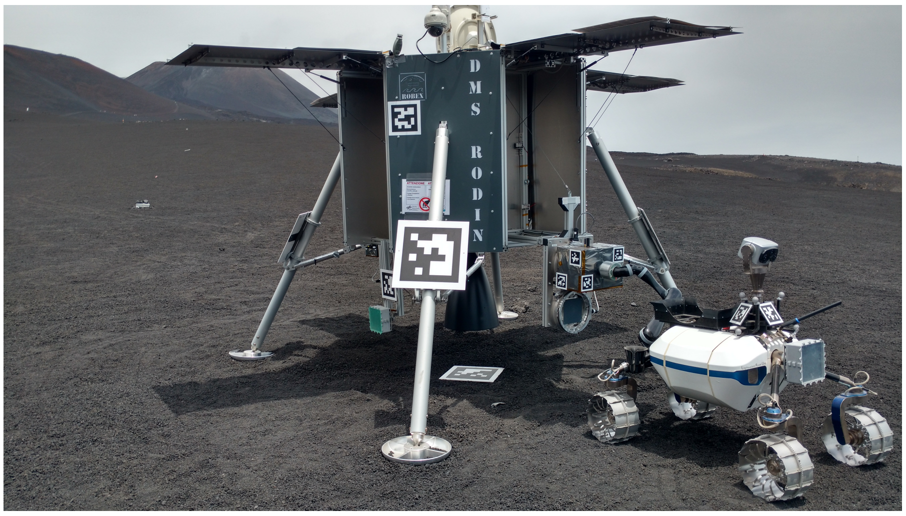


Figure 1.1.: The LRU2 collecting a payload box during a field test at Mt. Etna.©DLR

For the manipulation of tools and probes during planetary missions, where the rover's autonomy is of central importance, several constraints must be fulfilled [2]. According to [2] the planned manipulator motion must:

- be collision free,
- adhere to the manipulator's kinematic and dynamic constrains and
- accomplish the required task.

Hitherto, the LRU2 plans motions for its manipulator in an open loop fashion using predefined, object specific manipulation sequences, all following a similar pattern, as described in [1]:

1. detect manipulation object and estimate the object pose,
2. move the rover base to approach the object at predefined pose,
3. move the manipulator to approach the object at a subsequent predefined pose and
4. conclude the task by a sequence of relative impedance controlled movements.

While the rover takes advantage of on-board cameras to determine the current pose of its payload box, it so far relies solely on the measurements from its joint position encoders and the given kinematic model of its manipulator to calculate the end-effector pose.

As part of the Robotic Exploration of Extreme Environments (ROBEX) Alliance [3], the LRU2 took part in a moon-analogue mission [4] on Mt. Etna, in 2017. The mission showed that the lightweight robotic structure of both rover and arm leads to kinematic inaccuracies, which in turn result in imprecise end-effector positioning. Additionally, the system can be susceptible to extrinsic camera decalibration [5].

In fact, the inaccuracy of the kinematics of light-weight robotic arms is a well known problem. Both, the elasticity of the joints and small deviations from the computer-aided design (CAD) model, which is used as a basis for the definition of the arm's kinematic parameters, lead to an accumulated calibration error between end-effector and camera [6].

The docking interface of the LRU2's manipulator allows only for a limited positioning error of ± 1 cm, while the current kinematics exhibit inaccuracies above this threshold, depending on the manipulator's configuration. Because of this, in cases where elasticity induced joint displacements are large, the lack of feedback in such an open-loop control scheme can lead to failure of the docking procedure or mechanical damage of docking interface or probe. Even though the planned motion might be collision free and adhere to the given kinematic and dynamic constrains, the given task may not be accomplished.

Thus, it is desirable to introduce some form of feedback to the system, to prevent the aforementioned shortcomings. A solution to this problem is to include visual estimates of both end-effector and target pose in a closed-loop scheme, known as visual servoing. Its goal is the nullification of the pose error, based on the minimization of an error metric defined by features visible in the image stream [7] .

1.2. Objective

The objective of this thesis is to improve the accuracy and overall success rate of docking procedures between the rover's end-effector and payload boxes by incorporating visual pose estimates into the robot arm control loop. These estimates will be obtained by taking advantage of an active marker array consisting of 12 light-emitting diodes (LEDs), which is already mounted on the proximal face of the end-effector. In the following, the end-effector is also referred to as the docking interface.

Three sub-goals will be addressed for this purpose:

1. robust monocular tracking and pose estimation of the docking interface,

2. incorporation of propagated pose uncertainty for tracking noise reduction using an Error-State Kalman filter and
3. visual servoing of the LRU2 arm.

Since the LRU2 is designed for operation in outdoor environments, robustness of the tracking algorithm to challenging illumination conditions is paramount. However, as the rover's computational resources are limited, a light-weight tracking approach is needed. For this purpose, a monocular, active marker-based tracking method is proposed. It fuses measurements from vision and kinematics and ensures tracking robustness in varying lighting conditions.

During docking of end-effector and target box, two phases can be distinguished:
An *approach phase*, where the end-effector is positioned in front of the docking target, and a *docking phase* during which the rover establishes direct contact to the payload box. Thus, building on the tracking, two different visual servoing techniques will be implemented and tested for the control of the rover arm, accommodating the different requirements present in each of the two phases.

To ensure high positioning accuracy during the approach phase, a *position-based visual servo* will be implemented for this phase. To prevent damaging either the docking interface or the docking target, a stable interaction behavior must be ensured once end-effector and target object come into contact. Therefore, a *hybrid-impedance visual servo* strategy is proposed for docking maneuvers between the manipulator end-effector and a payload box. It is planned to maintain the correct end-effector position within the docking plane by means of the visual servo, while adopting an impedance based control strategy for the remaining degrees of freedom (DOF).

1.3. Thesis outline

In the following, we will begin by giving an overview over the existing rover system and its components in chapter 2.

The first part of chapter 3 will give an introduction to the topic of visual servoing and present related works in the field of vision-based manipulation and grasping, while the second part discusses popular tracking approaches used in visual servoing applications.

Chapter 4 gives a brief overview on theoretical and mathematical fundamentals that will be relevant throughout this work. Its focus lies on the fields of vision, control and probability theory.

In chapter 5, we will then dive deeper into the conception of the vision-based pose correction pipeline designed for the rover's payload docking procedure. For this purpose, we begin the chapter with an evaluation of challenges and system requirements and end with a presentation of the proposed system pipeline.

Building up on this, chapter 6 focuses on the implementation of the active marker tracking sub-system. It comprises the design of our proposed *gradient intersect tracking* method, followed by the description of the monocular pose estimation algorithm used for the retrieval of the docking interface's 3D pose. We further present an Error-State Extended Kalman filter formulation used for noise reduction and outlier removal.

Chapter 7 closes the gap between vision and control by presenting our chosen visual servo strategies for the approach and docking phase, respectively. It further includes the introduction of the simulation setup developed for the evaluation of the presented methods and discusses the evaluation results.

Lastly, chapter 8 concludes this thesis by summarizing its contents and achievements and providing an overview on open questions that arose over the course of this work.

2. System Overview

The following chapter introduces the rover and its components. This is followed by further details on the serial manipulator and its docking interface. Lastly, the available kinematic and vision data is discussed.

2.1. The LRU2

The LRU2 rover is a rover prototype designed for planetary exploration [8], where sensors and actors need to be able to work under alien conditions [8]. For this reason, it relies solely on sensor concepts which are known to work in such conditions and are currently deployed in space missions, such as stereo cameras and an inertial measurement unit [8].

To be able to collect and assemble objects, the rover features a six degree of freedom manipulator, the *Jaco2*[1]. It is mounted on the rear end of the rover with a base tilt of 45°, as visible in Fig. 1.1. The manipulator is equipped with a custom end effector, the *envicon docking interface* [2], designed specially for safe and efficient tool and payload handling [2]. A pan/tilt camera system is installed at the rover's top front and used for navigation and object detection. An additional stereo camera is mounted at its lower rear, at the base of the manipulator, and employed for object pose estimation during manipulation.

During the rover's construction, additional focus was set on its light-weight structure to demonstrate concepts to facilitate economic transportation to space for future planetary rovers [8]. The rover is 1090 mm long, 730 mm wide and weighs approximately 40 kg [8]. It is designed to drive through rough terrain at velocities up to 1.1 m/s. Its locomotion system is comprised of four individually steered and powered wheels which are connected to the main body by two bogie frames. These are mounted on the rover's front and rear, using serial elastic actuators (SEA) to assure both active and passive suspension in the bogies [8].

The main body houses the electronics needed to ensure the rover's autonomy. This includes batteries, an inertial measurement unit, and its computational unit. The latter is comprised of the main on-board computer, which is a standard industrial computer with an Intel Xeon E-2276ME CPU (2.80 GHz), an FPGA Spartan-6 Board used mainly for

[1]https://www.kinovarobotics.com/en/products/gen2-robot

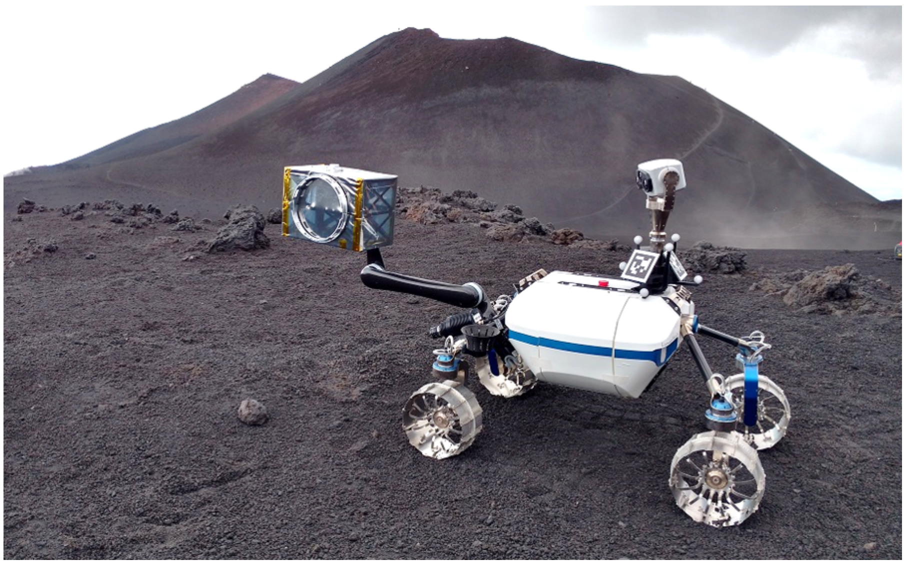


Figure 2.1.: The LRU2 during a field test at Mt. Etna, transporting a payload. ©DLR

the computationally expensive stereo processing, an Intel Atom E3845 CPU (1.91 GHz) running the real-time control loops of the manipulator and a BeagleBone Black computer board with an ARM Cortex-A8 processor used to address the manipulator through its RS485 interface [1]. It has to be pointed out that several computationally demanding tasks run on the main CPU in parallel, such as simultaneous localization and mapping, navigation, perception and manipulation, presenting the CPU with considerable process load.

To manage the interprocess communication between the various key components, the rovers software architecture makes use of several middlewares, such as the open-source *robot operating system (ROS)*. Further details on the rover platform can be found in [8].

2.1.1. The Jaco2 Arm

The Jaco2 arm is a light-weight robotic manipulator, which exhibits high dexterity within its workspace [1]. It is used for several purposes: The manipulation, deployment, and transportation of payload boxes, the collection of soil and rock samples, and the handling of tools. The manipulator weighs 4.4 kg, has a maximum reach of 90 cm and offers a maximum payload capacity of 2.2 kg when fully extended [9]. It has six rotational joints, each equipped with a DC brushless motor with Harmonic Drive technology featuring several sensors: encoders, accelerometers, current-, torque- and temperature sensors [10].

The original *Kinova Jaco2 API* has been replaced by a set of controllers designed and implemented at the RM to meet the control requirements, such as high positioning accuracy during tool handling and safe and robust contact behavior during object manipulation, needed throughout the rover's test missions [2]. Currently four different control modes are available:

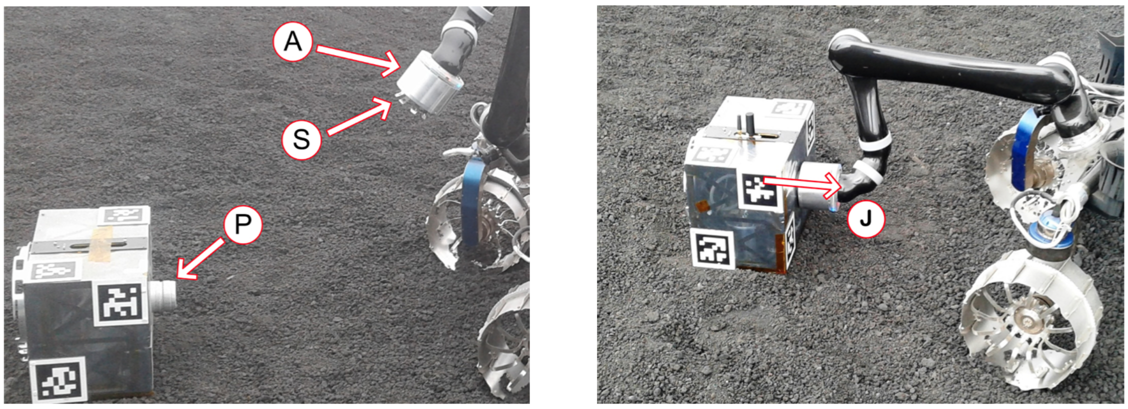
1. joint impedance control,
2. joint space position control,
3. Cartesian impedance control and
4. Cartesian position control.

Within this thesis, modes 2 and 3 will be discussed more in detail in chapter 7.

2.1.2. The Envicon Docking Interface

To ease the handling of tools and pick-up of payload boxes as well as to facilitate the data-, fluid-, and electrical power transfer, the original gripper was exchanged with the specially designed envicon docking interface. The interface consists of a passive and an active coupling partner, with the latter containing active metallic spring grasping elements [2], as seen in Fig. 2.2.

The passive coupling partner is a rotationally symmetric cylindrical structure with a tapered end section, that is mounted on the target objects.



(a) (b)

Figure 2.2.: a) Overview of the Envicon docking interface as given in [2]: Its passive part "P", denoted in the following as the docking target, and active part "A" containing spring grasping elements "S". "A" is also referred to as the end-effector. b) The docking is completed. "J" indicates the last joint of the manipulator. Figures taken from [2].

The active partner is a metallic, rotationally symmetric cylinder with an outer diameter of 102 mm, a length of 79 mm and an inner diameter of 65 mm. It weighs 390 g and can hold payloads of up to 5 kg [2]. Its inner part holds the docking core, which includes sensors, power supplies, the system controller unit and the grasping elements [2].

On its proximal end the cylinder is equipped with a circular array of 12 evenly distributed LEDs. Until now, these have been used for visual status communication, by changing between different color- and blinking modes. They are now used in a continuous light mode, to ensure their continuous visibility. Throughout this thesis, the LEDs will also be referred to as the *marker array* or *active marker array*.

2.2. Available Data

As stated in Sec. 2.1, the LRU2 is equipped with a number of different sensors to monitor the rover's state and its environment. These are, e.g., cameras supplying image streams of its surroundings, and encoders providing position measurements for the rover-arm joints.

The current configuration of the rover as well as its perceivable environment are stored in the form of a ROS transformation tree [11]. It contains the relative transformations, parameterized by the translation and quaternion based rotation, between two consecutive reference frames of the rover, or between the rover and objects within its environment. These are stored as a single, connected tree, enabling us to retrieve any relative transformation between two frames within the tree by a simple concatenation of relative transforms. The transformations can be distinguished into static and dynamic transformations. Each of the dynamic transformations is continuously updated at a rate between 10 Hz and 100 Hz, either by position encoders (robot joints) or information from the image stream (e.g., the payload box pose).

Altogether, the following hardware and data are directly available to us for tracking and servoing of the end-effector:

- an active marker array of 12 LEDs on the proximal face of the end-effector,
- gray-scale image data from the rover's pan/tilt stereo camera,
- RGB image data from the rover's bottom rear stereo camera,
- the manipulator's forward kinematics, given by the relative transformation between the arm base and the docking interface, attached to the manipulator's flange, within the transformation tree and
- payload box pose estimates, obtained by AprilTag detection, also accessible as part of the transformation tree.

Information on the camera's parameters, as well as their image streams, are available via individual ROS topics.

It has to be pointed out that even though the system provides us with two stereo cameras, stereo vision is not applicable in our case. Strong reflections, especially from the metallic docking interface, prevent us from obtaining usable depth estimates [12]. The use of a stereo setup is further discouraged by the limited computational resources and lack of a graphics processing unit that could be used to run more computationally expensive stereo vision algorithms on. For this reason, we have to limit ourselves to the use of monocular camera information. We choose to use the image data of the left top mounted pan/tilt camera for our active marker tracking pipeline. Though this camera offers only gray-scale images, thus excluding any color information for our process, it was chosen for two reasons: Firstly, it offers a higher update rate of 14 Hz compared to the 2 Hz provided by the base camera stream. Secondly, as we are planning to track a set of bright LEDs, it is desirable for the end-effector to have as little background light as possible from the observed perspective to ease the detection and tracking. Thus, the sky-wards view of the base camera is unfavorable in the context of outdoor operability. An additional point influencing our choice, was the uncertainty of whether or not the base camera would be removed in the context of major system upgrades, as preparation for the planned ARCHES Space-Analogue Demonstration Mission [13] on mount Etna.

3. State of the Art

In the following, we give an overview on the state-of-the-art for visual servoing and tracking. We begin this chapter with an introduction to visual servoing and present a selection of related works concerning the vision-based control of robotic arms for grasping and manipulation tasks. This is followed by an overview on tracking strategies, commonly encountered in visual servoing systems.

3.1. Visual Servoing: An Introduction

A visual servoing system is a closed-loop control system that uses visual information obtained by one or multiple cameras within the workspace as part of its feedback loop. It is used to ensure the autonomy of robotic systems in dynamic, unknown, or unstructured environments as well as in the presence of kinematic uncertainties. Example applications include minimally invasive surgery [14], guidance and control of unmanned vehicles [15] and autonomous on-orbit servicing of space robots for servicing, refueling, and docking of satellites [16].

Visual servoing has been an active research topic within the robotics community for several decades, with its first mention dating back to 1979 [17]. As a multidisciplinary research area in robotics, it combines different fields such as computer vision, kinematics, dynamics, real-time systems and control. In general, the objective of the servoing scheme is to bring the robotic end-effector into a desired pose (position and orientation) relative to a given target object. For this, depending on the camera configuration, either only the target or both end-effector and target have to be tracked by the vision system [7].

3.1.1. Classification

Visual servoing methods can be classified from several different perspectives. One of them is the type of camera configuration. Cameras used for visual servoing purposes are either stationarily fixed in the workspace, referred to as *eye-to-hand* configuration, or directly attached to the robot end-effector, also known as *eye-in-hand* configuration, see Fig. 3.1. In the eye-to-hand case, the camera is related to the manipulator base by a known transformation ${}^{C}\boldsymbol{T}_{B}$, whereas a second transformation ${}^{C}\boldsymbol{T}_{E}$ describes the dynamic, relative pose between the camera and end-effector. A further variation for the eye-to-hand case is given, when the camera is mounted on another moving base or pan/tilt head. This makes it possible to adjust its viewing angle and position relative to the end-effector, thus improving the vantage point if needed [7]. In the eye-in-hand case the transformation ${}^{C}\boldsymbol{T}_E$ is usually constant and known. Here, any movement of the robot directly induces camera motion [18].

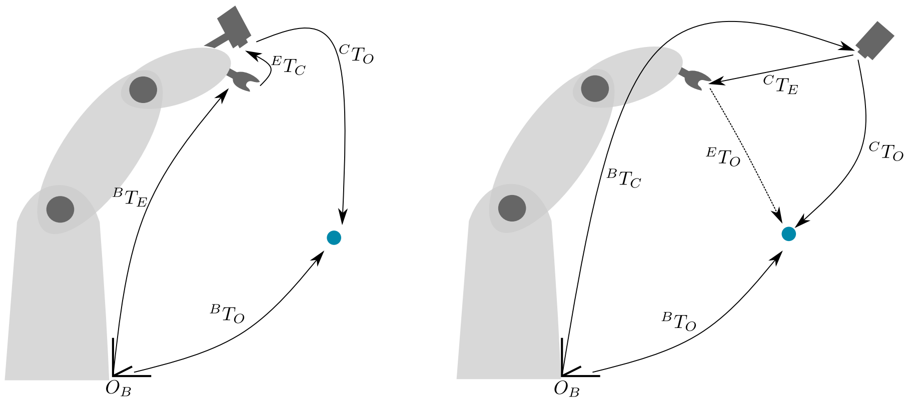


Figure 3.1.: Relevant transformations for different camera configurations. Left: Eye-in-hand case. Right: Eye-to-hand case. Shown are the robot, camera and object to be manipulated.

Employing a single camera minimizes the processing time needed to collect visual information. However, if the object model is unknown, loss of information about the objects depth along the optical axis complicates the control design [19]. Thus, monocular systems usually adopt some form of model based visual techniques to facilitate the estimation of the distance along the optical axis between the camera and the tracked object [19].

A further classification is based on the control input signal. Servoing schemes which eliminate the use of a separate robot controller entirely are known as *direct visual servo* controllers [20]. Direct visual servo controllers compute the torque inputs needed for the robot joints, without the need for a separate robot joint controller and stabilize the system using only the visual feedback [7].

Indirect, hierarchical control architectures on the other hand, which use the vision system to supply the actuating variables to joint level controllers, are referred to as *dynamic look-and-move* systems [7]. Hereby, the stability of the system relies on both the vision-based controller and robot joint controller. However, due to the low sampling rates of visual systems and the benefit of separating kinematic singularities from the vision component, most systems adopt the dynamic look-and-move approach [7].

Lastly, when it comes to the choice of the system error signal, two particular approaches can be distinguished: position-based visual servoing (PBVS) and image-based visual servoing (IBVS). Both are depicted in Fig. 3.2. In PBVS, also referred to as 3D visual servoing, image measurements of a calibrated camera are used together with a model of the target object to determine its relative pose to the camera [7]. While image plane features are observed during operation, both the object geometry and camera parameters need to be known in advance to retrieve the 3D pose. This is done using pose estimation algorithms based on image points [21–23], point and line correspondences [24], curves [25], or other geometrical features [26]. The error between either the camera (eye-in-hand case) or the observed end-effector pose (eye-to-hand case) and desired object pose is then given in Cartesian space and used within the control law to incrementally move the end-effector towards the goal object. It is important to note that PBVS strongly depends on the accuracy of the model of the object's geometry and on the correct camera calibration.

IBVS, also known as 2D visual servoing, provides the control input by operating on visual features directly in the image plane. Other than for PBVS an object model is not needed and no three dimensional pose is computed. The control problem is defined in terms of image coordinates, where the task is to move the coordinates of given feature points, e.g. dots of a fiducial marker, to their desired positions by moving the robot end-effector accordingly. This leads to reduced computational delay and eradicates errors otherwise induced by camera calibration. However, due to the highly non-linear relationship between image features and camera pose, IBVS is considered a challenging control problem [7].

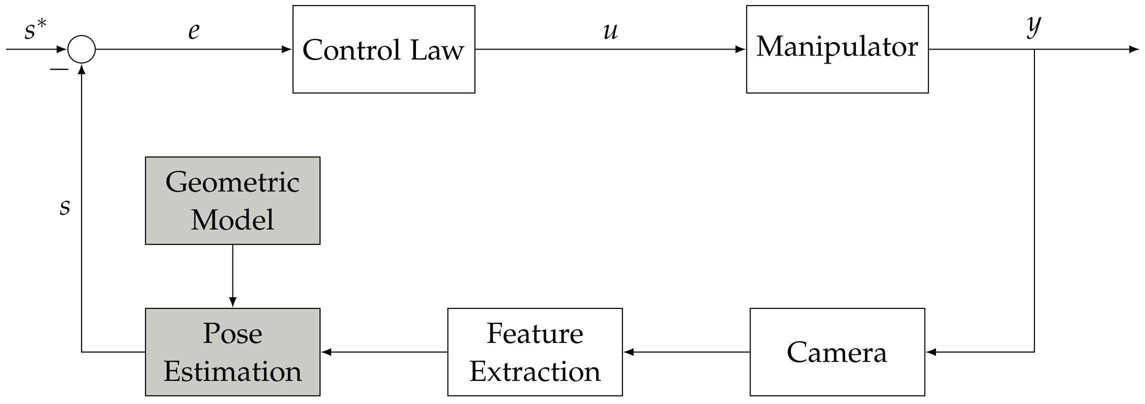


Figure 3.2.: A general block diagram for visual servoing. s^* and s are the desired and current feature coordinates in the image space (IBVS) or desired and current end-effector pose in Cartesian space (PBVS). Gray blocks apply only to PBVS. The control law generates a control input u (e.g. position or velocity) for the manipulator based on the feature error e. y marks the system output.

To alleviate some of the known issues of classical 2D and 3D visual servoing, several *hybrid visual servoing* approaches have been presented. The first hybrid visual servo, also called 2.5D visual servo, was introduced by Malis et. al. [27]. The idea behind it is to control the manipulator trajectory in both cartesian and image space at the same time. This can be done, e.g., by combining PBVS for the control of translational and IBVS for the control of the rotational DOF, respectively, to approximate straight line trajectories in both control spaces [27].

3.1.2. Related Work

Research on vision-based manipulation and grasping of objects with a robotic manipulator has been presented in several works.

Vahrenkamp et.al. [6] present a hybrid approach used to mitigate the effect of imprecise forward kinematics in light-weight robotic arms. For this purpose, the authors merge vision-based position estimates with orientation measurements obtained from the robot kinematics to control the hand of a humanoid robot during grasping and manipulation tasks. To deal with situations in which the hand tracking fails (e.g., due to strong reflections or other imaging artifacts), an approximation of the hand position is considered, taking into account the current end-effector position obtained by the kinematics and the last available difference between the pose obtained by the vision component and the kinematically calculated one.

In [28] a PBVS approach with alternated eye-to-hand/eye-in-hand configuration is proposed to improve the grasping of objects with a humanoid robot. The robot hand's pose is tracked by one of two monocular cameras using a set of planar markers. In the first stage, an eye-in-hand configuration is used to position the robot in front of the object, while the second stage utilizes an eye-to-hand configuration to steer the hand towards the target.

To ensure a compliant behavior and safe interaction on contact with the manipulation target, different works on hybrid impedance-based visual servoing strategies have been presented. Impedance controllers simulate the behavior of a mass-spring-damper system, regulating the relationship between the position of an end-effector and the forces exerted by it.

In [16], IBVS is used to position the manipulator of an on-orbit serving robot with an eye-in-hand camera configuration and a set of planar markers. Vision-based control is then coupled with impedance control to ensure the proper manipulation during the docking phase. The desired end-effector velocity calculated by the visual servo is thereby directly fed as an input to the impedance controller, together with the measured end-effector forces and torques. In several hardware-in-the-loop tests, the authors could show that the use of visual servoing not only during the approach, but also in the docking phase improves alignment and stability during impedance control and helps reducing interaction forces between the robot and the docking target.

In [29], a position-based visual impedance control scheme is proposed, using PBVS for movement in free space and extending it to *position-based visual impedance control* on contact. Further, to accurately estimate the target object pose, an Extended Kalman Filter is used to merge information from vision, joint and force measurements on contact.

3.2. Tracking for Visual Servoing

Video tracking is the process of automatically following previously detected moving objects or or points of interest (POI) over time, across a sequence of images. It is a vital part of many vision tasks, thus, an abundance of different tracking approaches have been presented to date. Those especially of interest for visual servoing purposes can be classified into three main categories: Feature-based [30, 31], model-based [32, 33] and marker-based [34–36] methods.

Feature-Based Tracking

Feature-based approaches track 2D features, such as points, corners, edges or contours directly in the image plane, without the need of a (geometric) model of the tracked object. They are thus especially well suited for use in IBVS methods.

Popular approaches for edge and contour tracking are M-estimators [37], used to robustly estimate the feature parameters using an iteratively reweighted least squares approach, and the moving-edges algorithm [38] intended to track parametric curves. The latter works by conducting a one-dimensional search in a restricted search interval along the normal of the contour points detected in the previous image.

Feature point based tracking requires the target object to provide trackable features. Especially corners and similar entities are considered good features to track as their strong spatial gradient yields robust information about the interframe motion [39]. The standard point feature tracking method used to date is the Kanade-Lucas-Tomasi (KLT) tracker [30], a differential tracker which is based on the difference in spatial intensity gradients of consecutive images. In the case of both, edges and keypoints being available, a combined hybrid tracker can be applied, e.g., using a Kalman filter to combine the two features [31], which increases the robustness by reducing error accumulation [33].

Even though feature-based tracking approaches are both simple and fast, they are impractical for our use case: Edge based methods prove to be sensitive to clutter [40], making them difficult to use in outdoor applications, while the application of feature point based methods is not ideal for the plain, metallic and textureless robotic end-effector used in this work.

Model-Based Tracking

Model-based methods rely on a 3D CAD model of the object for tracking and are closely related to the pose estimation problem. First, the CAD model is projected to the image plane. Then, image features such as keypoints on the visible part of the object surface or the objects contours found along the normal of the projected edges are extracted and put in correspondence with the model. Matched keypoints and/or contours are then used to estimate the objects 3D pose by minimizing the distance between the model and features in the observed image [41].

3D model-based methods have gained wide popularity in recent years. Their main advantage is an improved robustness compared to feature based methods by using knowledge available in the scene to predict observable object motions, e.g. in the cases when the tracked object is partially occluded. However, 3D model-based approaches require both high computational resources and dense depth information [5], none of which are available to us in the current setup of the rover.

Marker-Based Tracking

A popular choice in visual servoing applications are passive planar markers, such as white dots on a dark background, whose image coordinates can be tracked using simple real-time capable algorithms [42]. These types of markers are especially useful when it comes to the validation and testing of new visual servo control laws and modeling aspects [27, 43] as they can be tracked with high robustness and precision [27]. They can be used as references for the visual servoing application, either directly in the image plane or in 3D space by solving the Projective-n-Point (PnP) problem using their 2D/3D correspondences given by a geometric model of the marker.

More complex markers with a QR-code-like appearance, such as *AprilTags* [35], are used for the retrieval of the location and scale of tracked objects [44], pose estimation in augmented reality applications and the unique identification of different scene objects.

In general, marker-based tracking often corresponds to a repeated marker detection for every newly received frame. The detection can be based on a number of marker characteristics such as color, pattern or shape.

A special case of marker-based tracking is the tracking of so called *active* markers. Active markers can be patterns of LEDs radiating at predefined wavelengths in the visible or non-visible light spectrum or ball-shaped markers with a reflective coating. Use cases for active markers include virtual reality and medical applications. An example for the application of infrared beacons for head and hand pose applications is given in [36], while [45] track the pose of a miniature helicopter in flight using a set of LEDs, which are attached to its rotor tips and the end of the tail.

In conclusion, marker-based tracking is well suited for real-time tracking of simple predefined pattern and shapes, indicating the position and orientation of the tracked object. It requires less computational resources than the fitting of a complete 3D CAD model, making it applicable for use on systems with considerable process load, such as our rover.

4. Fundamentals

In the following, we introduce the fundamental concepts and nomenclature used within the remainder of this work. We start by giving an introduction to the image formation process, discussing important effects of lighting, such as reflections, affecting our use case. Subsequently, basic notions for the robotic system will be presented, such as different rotation representations used within later chapters and the conversion between them. Lastly, we will introduce the key concepts of probability theory and optimization, which will be of special interest in chapter 6.

4.1. Image formation

The image formation process describes the acquisition of 2D images from 3D objects. This encompasses two major fields. Firstly, the physics of light, which determines the appearance of a point in the image plane. Secondly, the geometry of image formation determining where on the image plane the projection of a point will be located.
This section gives a brief overview over both fields by summarizing the relevant photometric properties of the image formation process and introducing the camera model as well as its related nomenclature.

4.1.1. Photometry of Image Formation

The image of an object is not made up of 2D features, but instead defined as a two-dimensional function of pixels $\boldsymbol{x} = (x, y)$, where x and y indicate the row and column indices of the pixel, respectively. The amplitude $\boldsymbol{I}(\boldsymbol{x})$ for any pixel in the image is called the intensity or gray level of the image at that point [46, p.60]. It usually ranges between 0 for low and 255 for high (color-) intensity.

To capture the image of an object, it must be illuminated by either a point light source, e.g., a lamp, or an area light source, e.g. the sun [46, p.60]. The reflective properties of the illuminated object, in our case the manipulator and docking interface surfaces, hereby directly influence the quality of the retrieved imaging data.

For a surface with very small irregularities, compared to the lights wavelength, such as the polished metal of the our docking interface or the varnished synthetic parts covering the manipulator, specular reflections can occur. As a large proportion of the incoming light source is reflected, specular reflections can lead to *blind spots* of high intensity values within the object image, mimicking the appearance of the light source.

This makes it especially challenging to track active markers that are situated on or in the close vicinity of specularly reflective surfaces as the blind spots show very similar characteristics to the tracked LEDs.

4.1.2. Camera Model

The camera model describes the relationship of a 3D world point $\boldsymbol{x}_w = \begin{bmatrix} X_w & Y_w & Z_w \end{bmatrix}^T$ and its 2D image plane projection $\boldsymbol{x}_i = \begin{bmatrix} u & v \end{bmatrix}^T$, with u and v indicating the row and column of the image, respectively. It is of special importance when trying to recover the pose of a 3D object from a 2D image. In practice, these points are commonly described by their homogeneous representation, obtained by adding an additional entry containing a 1 to the end of the vector

$$\boldsymbol{x}_i = \begin{bmatrix} u & v & 1 \end{bmatrix}^T \qquad \boldsymbol{x}_w = \begin{bmatrix} X_w & Y_w & Z_w & 1 \end{bmatrix}^T . \tag{4.1}$$

The main advantage of employing homogeneous coordinates when working with projections, is the possibility of representing projective- and affine transformations by matrices [47], and thus simplify the calculations to matrix-vector operations. They further facilitate the description of points at infinity [47].

The Pinhole Camera

A simple, yet suitable approximation of how a scene is depicted by a camera with an ideal lens, is the *pinhole camera model*, depicted in Fig. 4.1. Its underlying assumption is that rays of light enter the camera through an infinitesimally small aperture, the pinhole. The model describes the mathematical relationship between a 3D point and its 2D projection by a perspective transformation

$$\boldsymbol{x}_i = \boldsymbol{K}\boldsymbol{P}\boldsymbol{x}_w \tag{4.2}$$

where $\boldsymbol{K}$ is the matrix of the so called camera intrinsic parameters

$$\boldsymbol{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.3}$$

containing the focal lengths f_x and f_y as well as the coordinates of the principle point (c_x, c_y). The parameters are determined in the process of camera calibration [48]. The principle point marks the intersection of the cameras principle axis and the image plane, while the focal length is a property of the camera lens. And $\mathbf{P}$ is the 3×4 matrix of extrinsic parameters, consisting of a rotation matrix $\boldsymbol{R}$ and a translation vector $\boldsymbol{t}$

$$\boldsymbol{P} = [\mathbf{R}|\mathbf{t}] \tag{4.4}$$

describing the cameras location, relative to the *world coordinates* of the captured object.

The overall perspective transformation is thus composed of two consecutive transformations. First, a homogeneous transformation maps the 3D world points $\boldsymbol{x}_w$ to the respective points in the camera frame $\mathbf{x_c} = \begin{bmatrix} X_c & Y_c & Z_c \end{bmatrix}^T$ using the extrinsic parameters. This is followed by a projective transformation projecting the points along the optical axis Z_c onto the image plane [47]. The image plane normalized coordinates are then given by $\boldsymbol{x}_{|c|} = [x, y, 1]^T$, with $x = X_c / Z_c$ and $y = Y_c / Z_c$ for $Z_c \neq 0$ [47].

Since the origin of the image coordinate system is usually not located at the principle point, applying the matrix of camera intrinsics to the point $\boldsymbol{x}_{|c|}$ shifts it to its homogeneous image coordinates

$$\boldsymbol{x}_i = \begin{bmatrix} u & v & 1 \end{bmatrix}^T = \begin{bmatrix} f_x x + c_x \\ f_y y + c_y \\ 1 \end{bmatrix} = \boldsymbol{K}\boldsymbol{x}_{|c|}, \tag{4.5}$$

accordingly. In our case, the origin is the left upper corner of the image. This can be rewritten in terms of pure pixel coordinates by omitting the third component, as $\boldsymbol{x}_i = \begin{bmatrix} u & v \end{bmatrix}^T$.

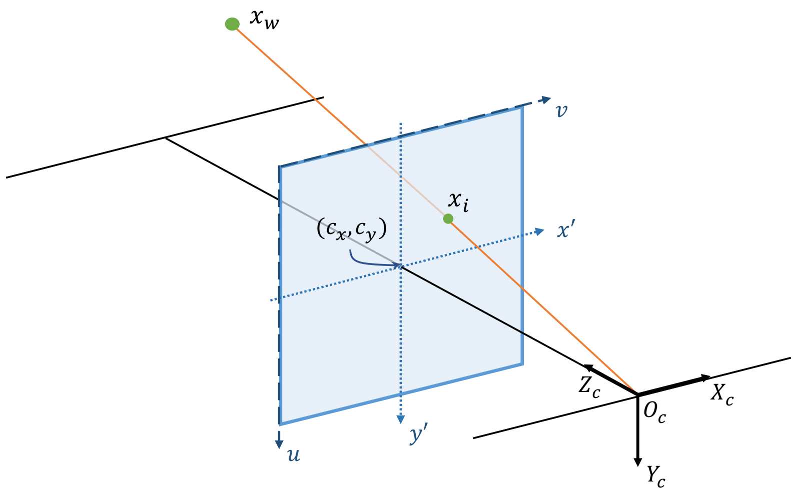


Figure 4.1.: Example of the pinhole camera model showing the projection $\boldsymbol{x}_i$ of a world point $\boldsymbol{x}_w$ onto the image plane. Also depicted are the camera frame with origin $\mathcal{O}_c$, principal point (c_x, c_y) and image coordinate system (u, v) as well as normalized camera coordinates (x, y). As seen in [49].

Extending the Pinhole Model

Two types of distortion effects can be distinguished: radial and tangential distortions. The pinhole model can be extended to account for those.

Several distortion models have been proposed in the literature. The distortion model applied by the LRU2 camera is the *Plumb Bob model*, proposed by Brown [50], which combines both distortion types. The model introduces five parameters $k_1, ..., k_5$ to correct for distortion effects.

4.2. Differential Geometry in Robotics

The following section gives an overview over the fundamentals of differential geometry in robotics. The focus hereby lies on serial manipulators without parallelisms or kinematic loops. For a more detailed and in-depth mathematical derivation of the presented concepts, the interested reader is referred to the corresponding literature [51, 52].

4.2.1. Robot Configuration

A serial robotic manipulator can be described as an open kinematic chain of rigid bodies, called links, where two consecutive links are connected by exactly one joint. These joints can be classified as being either rotary (e.g. revolute or spherical joints), translational (prismatic joints), or both (cylindrical joints). Commonly, serial manipulators are comprised exclusively of revolute joints [51]. Since the same holds for the manipulator used throughout this thesis, only rotational joints will be considered in the following.

A rotational joint is parameterized by a single rotation angle ϕ around its center axis. Thus, its minimal representation is given by the single parameter $q = \phi$ [52].
For a robotic manipulator its complete configuration defines the position of every of its joints. For a robot with n DOF the overall configuration space $\mathcal{C}$ is thus given by the vector

$$\boldsymbol{q} = \begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix} \in \mathbb{R}^n. \tag{4.6}$$

The configuration space of a robot is also referred to as *joint space* and the vector $\boldsymbol{q}$ is known as the *joint parameter vector*.

The last link of the kinematic chain is commonly referred to as *end-effector*. In our case this corresponds to the docking interface.

4.2.2. Rotation Representations

A rotation in 3D space is a length and angle preserving linear mapping, with three DOF, acting on a vector $\boldsymbol{r} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$. To formulate the rotation of an object, we can define a Cartesian coordinate system with orthonormal basis vectors $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3 \in \mathbb{R}^3$ attached to this object. The rotation is then expressed as a transformation acting on those basis vectors.

Rotation Matrices

The mapping to the new basis vectors $\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3 \in \mathbb{R}^3$ can be represented by a rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3\times3}$

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \tag{4.7}$$

The set of all rotational matrices forms the special orthogonal group $\mathbf{SO(3)}$, satisfying the constraints $\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}$ and $\det(\boldsymbol{R}) = +1$, with $\boldsymbol{I} \in \mathbb{R}^{3\times3}$ being the identity matrix [52].

A vector $\boldsymbol{p} \in \mathbb{R}^3$ given in frame $\boldsymbol{A}$ can be rotated into a new frame $\boldsymbol{B}$ using a rotation matrix ${}^B\boldsymbol{R}_A$ by a simple matrix-vector multiplication

$$\boldsymbol{p}_B = {}^B\boldsymbol{R}_A \boldsymbol{p}_A. \tag{4.8}$$

Multiple rotations expressed in this form can be concatenated by matrix-matrix multiplication. Rotation matrices are widely used to express rotations in robot kinematics as well as in computer vision, e.g. for camera calibration. However, with nine parameters used to describe the three DOF, rotation matrices are heavily over-parameterized and their imposed constraints make them unsuited for application in unconstrained optimization problems [53], e.g. for the iterative refinement of a visual object pose estimate.

Unit Quaternions

A popular alternative to rotation matrices is given by unit quaternions $\boldsymbol{q} \in \mathbf{SU}(2)$. Unit quaternions are parameterized by four values $\boldsymbol{q} = \begin{bmatrix} q_x & q_y & q_z & q_w \end{bmatrix}$, following the constraint $||\boldsymbol{q}|| = 1$ and can be represented as hypercomplex number [54]

$$\boldsymbol{q} = q_w + q_x i + q_y j + q_z k, \tag{4.9}$$

where q_x, q_y, q_z and q_w are real numbers and i, j, k are imaginary numbers for which $i^2 = j^2 = k^2 = ijk = -1$. q_w is also referred to as the real part while the vector

$\boldsymbol{q}_{im} = \begin{bmatrix} q_x & q_y & q_z \end{bmatrix}^T$ is the imaginary part. While all minimal rotation parametrizations, i.e. all representations containing exactly three parameters, suffer from singularities [55, p.14], quaternions are the representation with the fewest parameters that is still free of such issues.

Two quaternions $\boldsymbol{q}$ and $\boldsymbol{g}$ can be concatenated by a quaternion multiplication [54]

$$\boldsymbol{q} \otimes \boldsymbol{g} = \begin{bmatrix} q_w g_w - \boldsymbol{q}_{im} \boldsymbol{g}_{im} \\ q_w \boldsymbol{g}_{im} + g_w \boldsymbol{q}_{im} - \boldsymbol{q}_{im} \boldsymbol{g}_{im} \end{bmatrix}. \tag{4.10}$$

An arbitrary vector $\boldsymbol{v} \in \mathbb{R}^3$, written as a quaternion $v = \begin{bmatrix} v_1 & v_2 & v_3 & 0 \end{bmatrix}$ with a zero real part, can be rotated around an axis $\hat{\boldsymbol{n}}$ by an angle θ_q using a quaternion with [54]

$$q_w = \cos(\frac{\theta_q}{2}) \qquad \boldsymbol{q}_{im} = \hat{\boldsymbol{n}} \sin(\frac{\theta_q}{2}), \tag{4.11}$$

and its conjugate quaternion $\bar{\boldsymbol{q}} = \begin{bmatrix} -\boldsymbol{q}_{im} & q_w \end{bmatrix}$ by a quaternion-vector multiplication

$$\boldsymbol{q}(\boldsymbol{v}) = \boldsymbol{q}\boldsymbol{v}\bar{\boldsymbol{q}}. \tag{4.12}$$

Lastly, a unit quaternion can be converted into its corresponding rotation matrix with [56]

$$\boldsymbol{R} = \begin{bmatrix} 1 - 2q_y^2 - 2q_z^2 & 2q_x q_y - 2q_z q_w & 2q_x q_z + 2q_y q_w \\ 2q_x q_y + 2q_z q_w & 1 - 2q_x^1 - 2q_z^2 & 2q_y q_z - 2q_x q_w \\ 2q_x q_z - 2q_y q_w & 2q_y q_z + 2q_x q_w & 1 - 2q_x^2 - 2q_y^2 \end{bmatrix}. \tag{4.13}$$

Axis/Angle Representation

The last rotation parameterization to be discussed here is the axis/angle representation. An arbitrary rotation matrix can be represented by a rotation angle θ around a single axis $\boldsymbol{a} \in \mathbb{R}^3$. It is important to note, that only the direction of $\boldsymbol{a}$ is of importance, reducing its DOF to two. Both angle and rotation axis can thus be combined into a single vector $\boldsymbol{\omega}$, making this parameterization well suited for use in constraint optimization problems. The vector direction hereby gives the rotation axis $\boldsymbol{a}$, while its length corresponds to the rotation angle θ [53]

$$\boldsymbol{a} = \frac{\boldsymbol{\omega}}{||\boldsymbol{\omega}||} \qquad \theta = ||\boldsymbol{\omega}||. \tag{4.14}$$

By using the skew-symmetric matrix $[\boldsymbol{\omega}]_\times$

$$[\boldsymbol{\omega}]_\times = \begin{bmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} \end{bmatrix}_\times = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{4.15}$$

its corresponding rotation matrix can be calculated by using the exponential map from $\mathfrak{so}(3)$ to $\mathbf{SO}(3)$ [57]

$$\boldsymbol{R} = e^{[\omega]_\times} = \mathbf{I} + \frac{\sin\theta}{\theta}[\omega]_\times + \frac{1-\cos\theta}{\theta^2}[\omega]_\times^2, \tag{4.16}$$

with $\mathbf{I} \in \mathbb{R}^{3\times 3}$ being the identity matrix, while the axis/angle representation can in turn be obtained from a rotation matrix with [51]

$$\theta = \cos^{-1}\left(\tfrac{r_{11}+r_{22}+r_{33}-1}{2}\right) \qquad \boldsymbol{a} = \frac{1}{2\sin\theta}\begin{pmatrix} r_{32}-r_{23} \\ r_{13}-r_{31} \\ r_{21}-r_{12}\end{pmatrix}. \tag{4.17}$$

The log map from $\mathbf{SO}(3)$ to $\mathfrak{so}(3)$ is also given by

$$\boldsymbol{\omega} = \log(\boldsymbol{R}) = \frac{\theta}{2\sin\theta}\left[\boldsymbol{R} - \boldsymbol{R}^T\right] \text{ for } \theta \neq 0. \tag{4.18}$$

The axis/angle parameterization is especially useful for the representation of angular velocities by replacing the angle θ with the the rate of rotation $\Delta\theta\Delta t$ around the instantaneous axis of rotation $\boldsymbol{a}$. Further details on the transformation between rotation representations can be found in [55, p.14 ff].

4.2.3. Rigid Body Transformations

To plan movements of the end-effector, knowledge about its pose is of particular importance. The pose of a rigid body is given by a rigid body transformation defining the position and orientation of its body coordinate system relative to another body or reference frame.

Rigid body transformations include both a rotation $\boldsymbol{R} \in \boldsymbol{SO}(3)$ and translation $\boldsymbol{t} \in \mathbb{R}^3$ resulting in a function $\boldsymbol{T} : \mathbb{R}^3 \to \mathbb{R}^3$. When applying rigid transformations, the distance between each pair of points as well as the orientation between pairs of vectors are maintained. With the usage of homogeneous coordinates a rigid transformation can easily be expressed as matrix

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{4.19}$$

The transform of a point $\boldsymbol{x} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$, written in homogeneous coordinates, from one frame of reference $\boldsymbol{A}$ into an other frame $\boldsymbol{B}$ then defaults to a simple matrix vector multiplication

$$\begin{aligned} \boldsymbol{x}_B &= {}^B\boldsymbol{T}_A \boldsymbol{x}_A \\ &= {}^B\boldsymbol{R}_A \boldsymbol{x}_A + {}^B\boldsymbol{t}_A. \end{aligned} \tag{4.20}$$

4.2.4. Kinematics

As stated in Sec.4.2.1, the robots joint space is described through a set of joint parameters. These parameters are ultimately used to set the inputs for low level control of the manipulator. However, the manipulator's tasks are more intuitively described within the m dimensional *task space* $\mathcal{T}$, where m denotes the number of DOF needed to complete the task. For our case, the task space corresponds to the 3D Cartesian space, yielding $m = 6$ DOF. These are given by the position $\boldsymbol{p} \in \mathbb{R}^3$ and minimal representation for the orientation $\boldsymbol{\theta} \in \mathbb{R}^3$ of the *tool frame* E located at the end-effector. Its pose relative to a reference frame B is thus

$$^{B}\boldsymbol{x}_E = \begin{bmatrix} ^{B}\boldsymbol{p}_E \\ ^{B}\boldsymbol{\theta}_E. \end{bmatrix} \tag{4.21}$$

Consequently, to map between joint and task space, appropriate mapping functions are needed.

Forward Kinematics

The forward kinematics is concerned with finding the pose of the end-effector relative to the manipulator base [51, p.28].

Definition 1. *Let $q \in \mathcal{C}$ and $x \in \mathcal{T}$. The function $f : \mathcal{C} \to \mathcal{T}$*

$$^{B}\boldsymbol{x}_E = \boldsymbol{f}(\boldsymbol{q}) \tag{4.22}$$

that maps the joint space coordinates into their respective task space is called forward kinematics.

With knowledge of the current joint parameter vector as well as the geometric parameters describing the robot links, the forward kinematic problem can be solved by simply concatenating the rigid body transformations between adjacent links along the manipulator chain [51]:

$$^{B}\boldsymbol{T}_E = ^{B}\boldsymbol{T}_1^1\boldsymbol{T}_2 \ldots ^{E-1}\boldsymbol{T}_E, \tag{4.23}$$

where the indices B and e denote the base frame and tool frame respectively.

Inverse Kinematics

The inverse kinematics instead searches for the robot joint parameters needed to reach a specified end-effector pose in the task space

$$\boldsymbol{q} = \boldsymbol{f}^{-1}(^{B}\boldsymbol{x}_E) \tag{4.24}$$

Different approaches exist to the solution of the inverse kinematics problem [55]. In the following, we will focus on the numerical approach which uses the instantaneous kinematics to relate the task space and joint space velocities to each other and retrieve the absolute joint space parameters by integration.

Instantaneous Kinematics

While the forward kinematics describes only the static configuration of a manipulator, the instantaneous kinematics formulates the relationship between the vector of angular velocity of the joints $\dot{\boldsymbol{q}}$ of the manipulator and the resulting velocity $\dot{\boldsymbol{x}}_E$ of the end-effector in the task space

$$^{B}\dot{\boldsymbol{x}}_E = {}^{B}\boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{4.25}$$

by introducing the body Jacobian $^{B}\boldsymbol{J}(\boldsymbol{q})$, a square, symmetric matrix containing all partial derivatives of the forward kinematics function

$$^{B}\boldsymbol{J}(\boldsymbol{q}) = \left[\frac{\delta f_i}{\delta q_i}\right] = \begin{bmatrix} \frac{\delta f_1}{\delta q_1} & \cdots & \frac{\delta f_1}{\delta q_n} \\ \vdots & \ddots & \vdots \\ \frac{\delta f_m}{\delta q_1} & \cdots & \frac{\delta f_m}{\delta q_n} \end{bmatrix} \in \mathbb{R}^{m \times n}, \tag{4.26}$$

where m is the dimension of the task space and n the number of joints. For simplicity, we will write $^{B}\boldsymbol{J}(\boldsymbol{q})$ as $^{B}\boldsymbol{J}$ in the following.

If $m = n$, we can obtain a solution to the inverse kinematics by solving

$$\dot{\boldsymbol{q}} = {}^{B}\boldsymbol{J}^{-1B}\dot{\boldsymbol{x}}_E \tag{4.27}$$

and find the joint parameters by integration [55]. However, a problem arises, when the manipulator approaches a singular configuration. In a singularity, the robot loses one or more DOF of its output motion, as two or more columns of the Jacobian matrix become linearly dependent [55]. Formally, a singular configuration is defined in terms of the determinant of the Jacobian. A joint configuration $\boldsymbol{q}$ is called singular with respect to the task space coordinates, if

$$\det {}^{B}\boldsymbol{J} = 0\,. \tag{4.28}$$

The Jacobian then loses its square form and its inverse no longer exists. To avoid this problem, the Jacobian inverse in Eq. 4.27 can be replaced by its Moore-Penrose pseudo-inverse $^{B}\boldsymbol{J}^{+}$. The pseudo-inverse is defined as

$$^{B}\boldsymbol{J}^{+} = {}^{B}\boldsymbol{J}^{\top}({}^{B}\boldsymbol{J}{}^{B}\boldsymbol{J}^{\top})^{-1} \tag{4.29}$$

and allows finding the best possible solution to the inverse problem in the least squares sense [58].

4.3. Probability Theory and Optimization

In real-world applications, sensor readings, such as camera images and position encoders, are generally afflicted by a certain amount of noise, leading to uncertainties in their measurements. When considering its uncertainty, a measurement can be treated as a *random variable*. The following gives an overview on key concepts of probability theory needed to understand the error description and propagation used in the context of parameter estimation from image measurements throughout this thesis.

4.3.1. Scalar Random Variables

A random variable $\boldsymbol{X}$ can be interpreted as a mapping from a set of expected results of an experiment to a set of real numbers [59]. A measure of how much we we can expect the random variable to vary from the mean is its variance, defined as [59, p.56]:

$$\begin{aligned}\sigma^2 &= \mathbb{E}[(X-\bar{x})^2] \\ &= \mathbb{E}[X^2 - 2X\bar{x} + \bar{x}^2] \\ &= \mathbb{E}(X^2) - 2\bar{x}^2 + \bar{x}^2 \\ &= \mathbb{E}(X^2) - \bar{x}^2.\end{aligned} \tag{4.30}$$

We can now use the notation $X\ (\bar{x}, \sigma^2)$ to describe the random variable by its mean and variance.

The relationship between multiple random variables, such as X and Z, can be described by the covariance

$$\begin{aligned}C_{XZ} &= \mathbb{E}[(X-\bar{x})(Z-\bar{z})] \\ &= \mathbb{E}[(XZ) - \bar{x}\bar{z}]\end{aligned} \tag{4.31}$$

deviation from the mean for both variables [59, p.62].

4.3.2. Multivariate Random Variables

Random variables are not limited to be scalar but can take a vector form, in which case their associated quantities are also given in the form of vectors or matrices.

Let X and Z be two random variables with n and m elements, respectively, both given as column vector. Their covariance is then defined as [59, p. 66]:

$$\begin{aligned}\boldsymbol{\Sigma}_{XZ} &= \mathbb{E}[(X-\bar{X})(Z-\bar{Z})^T] \\ &= \mathbb{E}(XZ^T) - \bar{X}\bar{Z}^T.\end{aligned} \tag{4.32}$$

4.3.3. Measurement Formation and Likelihood

Suppose $\boldsymbol{y} \in \mathbb{R}^d$ is a vector containing the measurement or observation of a set of true underlying state values $\boldsymbol{\theta} = (\theta_1, ..., \theta_d)$ affected by noise $\boldsymbol{\epsilon}$. The noise can in most cases be assumed to be zero-mean Gaussian noise, referred to as white noise, with covariance

$\mathbf{\Sigma}$. Treating $\boldsymbol{Y} = (\boldsymbol{y}_1, ..., \boldsymbol{y}_N)$ as a set of N independent random variables, we can write for each observation $i = 0 \dots N$ [56, p. 29]:

$$\boldsymbol{y}_i = \boldsymbol{g}_i(\boldsymbol{\theta}) + \boldsymbol{\epsilon}_i \qquad\qquad \boldsymbol{\epsilon}_i \sim \mathcal{N}(\boldsymbol{\epsilon}_i | 0, \Sigma_i) \tag{4.33}$$

where $\boldsymbol{g}_i$ is the measurement function of the state variable.

This implies that the observation follows a *Gaussian distribution* of the form

$$\begin{aligned} p(\boldsymbol{y}_i | \boldsymbol{\theta}, \Sigma_i) &= \mathcal{N}(\boldsymbol{y}_i | \boldsymbol{\theta}, \Sigma_i) \\ &= \frac{1}{2\pi^{d/2} |\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2} (\boldsymbol{y}_i - \boldsymbol{g}_i(\boldsymbol{\theta}))^T \Sigma_i^{-1} (\boldsymbol{y}_i - \boldsymbol{g}_i(\boldsymbol{\theta})) \right). \end{aligned} \tag{4.34}$$

with p being the probability density function.

Applying *maximum likelihood estimation* we can find the state estimate $\boldsymbol{\theta}_{ML}$ that best explains our observations.

The overall *likelihood* function is then defined as [60, Ch. 2]

$$\boldsymbol{L}(\boldsymbol{y}, \boldsymbol{\theta}) = \prod_{i=1}^{N} p(\boldsymbol{y}_i | \boldsymbol{\theta}, \Sigma_i). \tag{4.35}$$

Using the logarithm, which is a monotonic function, we can turn the product into a sum while preserving the original maximizer. This yields the *log-likelihood* function [60, Ch. 2]

$$\log \boldsymbol{L}(\boldsymbol{y}, \boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\boldsymbol{y}_i | \boldsymbol{\theta}, \Sigma_i) \tag{4.36}$$

which we can now use to solve for the maximum likelihood estimate of the state

$$\boldsymbol{\theta}_{ML} = \arg\max_{\theta} \log \boldsymbol{L}(\boldsymbol{y}, \boldsymbol{\theta}). \tag{4.37}$$

Since maximizing the log-likelihood is the same as minimizing the negative log-likelihood, using Eq. 4.33 and neglecting constant terms we can write

$$\begin{aligned} \boldsymbol{\theta}_{ML} &= \arg\max_{\theta} \sum_{i=1}^{N} \log p(\boldsymbol{y}_i | \boldsymbol{\theta}, \Sigma_i) \\ &= \arg\max_{\theta} \sum_{i=1}^{N} \log \left(\frac{1}{2\pi^{d/2} |\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2} (\boldsymbol{y}_i - \boldsymbol{g}_i(\boldsymbol{\theta}))^T \Sigma_i^{-1} (\boldsymbol{y}_i - \boldsymbol{g}_i(\boldsymbol{\theta})) \right) \right) \\ &= \arg\min_{\theta} \sum_{i=1}^{N} \left(\frac{1}{2} (\boldsymbol{y}_i - \boldsymbol{g}_i(\boldsymbol{\theta}))^T \Sigma_i^{-1} (\boldsymbol{y}_i - \boldsymbol{g}_i(\boldsymbol{\theta})) \right), \end{aligned} \tag{4.38}$$

yielding the well-known *least-squares* problem [56, p. 30].
More specifically, if $\boldsymbol{g}(\cdot)$ is a linear function, Eq. 4.38 is known as the *weighed-least-squares* problem and a closed form solution can be obtained using e.g. singular value

decomposition. However, if $\boldsymbol{g}(\cdot)$ is non-linear, generally no closed form solution exists and iterative methods have to be applied instead [56]. In the following, two such methods will be presented, namely the Gauss-Newton and Levenberg-Marquart methods.

4.3.4. Iterative Methods

Assuming $\boldsymbol{\Sigma}$ to be a diagonal matrix, Eq. 4.38 can be rewritten as [56]

$$\boldsymbol{\theta}_{ML} = \arg\min_{\theta} \sum_{i=1}^{N} \left(\frac{1}{2} \frac{(\boldsymbol{y}_i - \boldsymbol{g}_i(\boldsymbol{\theta}))^T (\boldsymbol{y}_i - \boldsymbol{g}_i(\boldsymbol{\theta}))}{\sigma_i^2} \right), \tag{4.39}$$

where the term $\boldsymbol{r}_i(\theta) := \left(\frac{y_i - g_i(\theta)}{\sigma_i}\right)$ is commonly referred to as a residual. All residuals can be stacked to form a vector $\boldsymbol{r}(\theta)$. We are thus interested in finding the parameter vector θ that minimizes $||\boldsymbol{r}(\theta)||^2$ and can rewrite Eq. 4.39 as [56]

$$\boldsymbol{\theta}_{ML} = \arg\min_{\theta} \boldsymbol{E}(\boldsymbol{\theta}) = \arg\min_{\theta} \frac{1}{2} \boldsymbol{r}(\boldsymbol{\theta})^T \boldsymbol{r}(\boldsymbol{\theta}). \tag{4.40}$$

Gauss-Newton Method

In the case of nonlinear residuals, the Gauss-Newton method is based on the linear approximation using their first Taylor expansion in a small neighborhood $\boldsymbol{h}$ of $\boldsymbol{\theta}$ [61]

$$\boldsymbol{r}(\boldsymbol{\theta} + \boldsymbol{h}) \approx \boldsymbol{r}(\boldsymbol{\theta}) + \boldsymbol{J}_r^T \boldsymbol{h}, \tag{4.41}$$

with $(\boldsymbol{J}_r)_{ij} = \frac{\partial r_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_j}$ being the Jacobian of $\boldsymbol{r}(\boldsymbol{\theta})$.

Inserting this into Eq. 4.40 the approximated error is given by

$$\boldsymbol{E}(\boldsymbol{\theta} + \boldsymbol{h}) = \frac{1}{2} \boldsymbol{r}(\boldsymbol{\theta})^T \boldsymbol{r}(\boldsymbol{\theta}) + \boldsymbol{h}^T \boldsymbol{J}_r^T \boldsymbol{r}(\boldsymbol{\theta}) + \frac{1}{2} \boldsymbol{h}^T \boldsymbol{J}_r^T \boldsymbol{J}_r \boldsymbol{h}. \tag{4.42}$$

Starting with an initial guess for the minimum $\boldsymbol{\theta}^0$ the Gauss-Newton methods updates it iteratively by finding the step $\boldsymbol{h}_{GN}$ that further minimizes the error by setting the derivative of $\boldsymbol{E}(\boldsymbol{\theta} + \boldsymbol{h})$ equal to zero and solving for

$$\boldsymbol{h}_{GN} = -(\boldsymbol{J}_r^T \boldsymbol{J}_r)^{-1} \boldsymbol{J}_r^T \boldsymbol{r}(\boldsymbol{\theta}). \tag{4.43}$$

The new $\boldsymbol{\theta}^k$ is then given by

$$\boldsymbol{\theta}^k := \boldsymbol{\theta}^{k-1} + \boldsymbol{h}_{GN} \tag{4.44}$$

and the procedure is repeated until $\boldsymbol{h}_{GN}$ is sufficiently small. In the case of good initialization of $\boldsymbol{\theta}$, the approximation shows quadratic convergence [61]. However, it is important to note that in the case of bad initialization or $\boldsymbol{J}_r$ having a non-trivial nullspace, convergence of the algorithm cannot be assured [61].
A method that counteracts these problems is the Levenberg-Marquardt method.

Levenberg-Marquardt Method

The Levenberg-Marquard method can be described as a *damped* Gauss-Newton method [61]. By adding a damping factor λ to the calculation of the update step is expressed as

$$\boldsymbol{h}_{LM} = -(\boldsymbol{J}_r^T \boldsymbol{J}_r + \lambda \boldsymbol{I})^{-1} \boldsymbol{J}_r^T \boldsymbol{r}(\boldsymbol{\theta}) \tag{4.45}$$

with $\boldsymbol{I}$ being the identity matrix. One of the effects of this extension is that $(\boldsymbol{J}_r^T \boldsymbol{J}_r + \lambda \boldsymbol{I})$ is now guaranteed to be positive definite for all $\lambda > 0$ and thus always invertible. For large λ the Levenberg-Marquard method corresponds to performing gradient descent, while equaling the Gauss-Newton update when $\lambda \to 0$ [61].

5. Conception

In the following, we derive the concept for the vision-based payload docking pipeline. For this purpose we begin this chapter with the identification of system challenges.

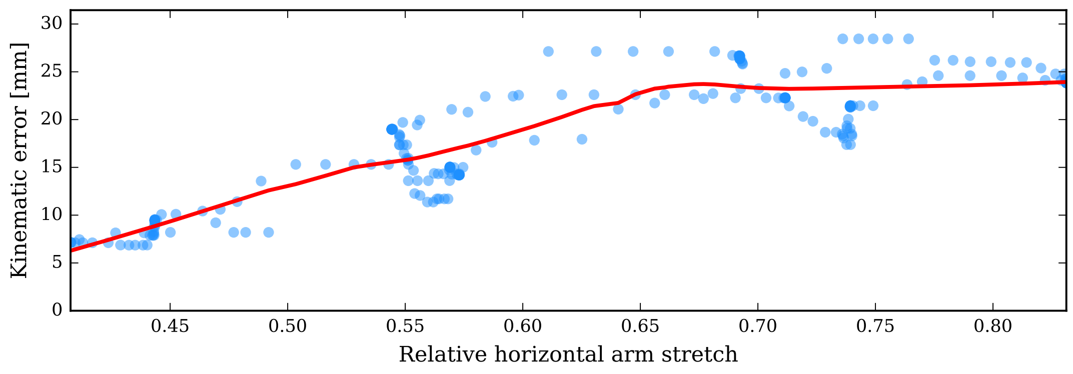
5.1. Error Analysis and Challenges

The rover system presents us with several challenges. Most notably, the imprecise manipulator forward kinematics, that motivated the introduction of vision-based feedback in the first place. The lightweight structure of the manipulator comes with a higher flexibility in the joints compared to industrial robots [62], which in turn leads to additional joint displacements that are not captured by the encoder measurements. These extra displacements lead to erroneous frame-to-frame transformations along the manipulator chain. The overall kinematic error at the end-effector is the sum of all transformation errors along the chain. If there was no kinematic error, the observed flange frame would coincide with the reference frame of the active marker array, situated in the center of the plane spanned by the LED circle.

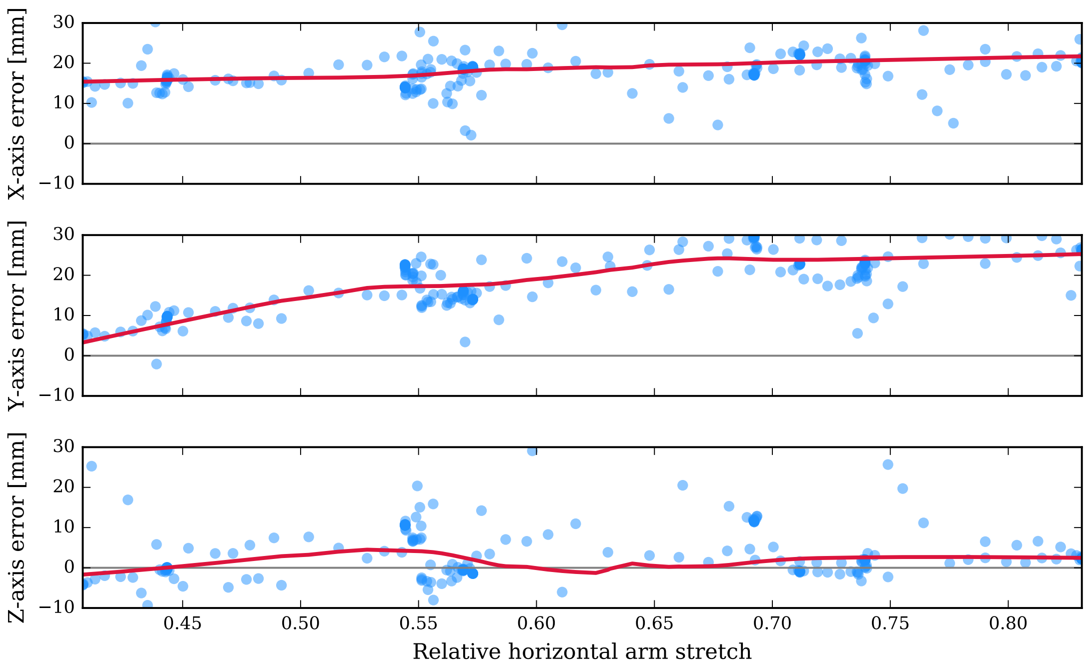
For the analysis of the dynamic error development we recorded a data-set which includes ground-truth measurements of the end-effector position, obtained by a Vicon® tracking system. Fig. 5.1a shows the relative kinematic error of the end-effector position in relation to the horizontal stretch of the arm. Unfortunately, sporadic delays in the ground-truth measurements led to strong deviations in the kinematic error, which do not reflect the real error characteristics of the system. We have therefore applied locally weighted scatterplot smoothing (LOWESS) with a point fraction of 0.5 to the presented figure, to illustrate an approximation of the real error to stretch characteristics.

It is important to note that the total error is not constant, but depends on the current arm configuration. If the arm is completely stretched out, the total end-effector displacement error will be minimal when the arm's longitudinal axis is perpendicular to the ground, while it will be maximal when aligned in parallel to it. The reason for this is that as the arm aligns with the ground plane, the lever increases and gravity leads to a larger displacement of the joints.

Fig. 5.1b further depicts the development of the position error, as given by the LOWESS filtered measurements, broken down for the three individual axes of the flange frame measurements, relative to the camera reference frame. The Z component hereby in-



(a)



(b)

Figure 5.1.: a) Trend (red line) of the normed relative error of the end-effector position, as obtained by the kinematics (blue dots), with regard to the relative horizontal stretch of the arm. b) The development of the position error from (a) broken down to the indivisual errors along the X, Y and Z axes of the camera reference frame. Note that the Z-axis corresponds to the optical axis of the camera, which is kept horizontal to the ground plane.

dicates the optical axis of the camera, which is approximately parallel to the ground plane. Additionally, note that the transformation between the manipulator base and the pan/tilt camera frame is kept constant during the time of recording the data-set. It confirms that the error is largest in those DOF most affected by gravity, while the error along the optical axis stays minimal. Nonetheless, the knowledge about the approximate end-effector pose presents to be a valuable source of information that can be used to facilitate the visual tracking and detection of the docking interface within the image stream.

The expected outdoor operability of the rover poses a further challenge. As an exploration rover prototype, the LRU2 is used to evaluate possible software and hardware concepts for real space rovers and thus needs to be able to operate robustly in outdoor environments exhibiting varying illumination conditions. Our choice of using the pan/tilt camera's gray-scale image stream makes it particularly demanding to track the active marker array, as it becomes difficult to distinguish the LEDs from the bright image background, as visible in Fig. 5.2.

The exterior of the robot arm is made of smooth, black plastic, while the docking interface consists of polished metal. The glossiness of both surfaces further complicates the robust visual tracking of the marker points, as it leads to specular reflections, displaying a similar gray-scale value to that of the LEDs. In our application we are confronted with two different types of specular reflections affecting the tracking of the end-effector: Strong reflections on the metallic docking interface in the immediate vicinity of one or more marker points affect the appearance of the LED in the image, making it difficult to identify them as such. Secondly, LED-like reflections on the manipulator surface can be easily mistaken for real LEDs, leading to outliers in the image data.

Lastly, we have to consider the differing demands for the chosen control strategies of the manipulator, depending on whether the arm is only approaching the docking target, without having established any direct contact with it yet, or is actively docking. In the former case, the focus lies on the correct positioning and alignment of the end-effector in front of the docking target. In the later, it is desirable to adopt some form of hybrid compliant control strategy that takes into account the forces acting on both the docking interface and the docking target, as not to damage either one of them and to ensure the docking procedure can be completed safely.

In addition, we are also presented with a set of unforeseen challenges introduced by the ongoing SARS-CoV-2 pandemic. With student access to the institute being strongly restricted and even prohibited for several months, it was unpredictable whether testing and evaluation of the presented concepts could be conducted on the real rover as planned. During these months, the rover further received several hardware and software updates, which had been delayed due to the pandemic. It is still unknown at the point of writing, when the rover will be back to its full and stable working state.

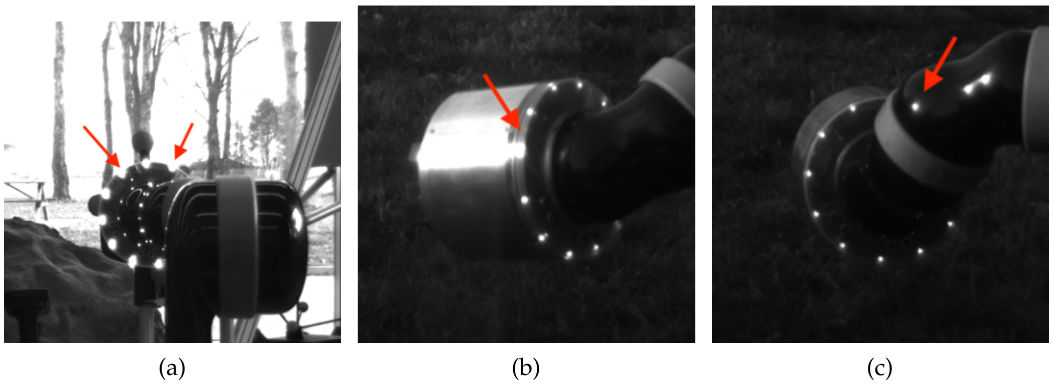


Figure 5.2.: The marker array as seen under varying back light and with different reflective artifacts. (a) illustrates the effect of strong back light to the identification of the marker points in the image. The arrows indicate LEDs whose gray values are indistinguishable from the background. (b) shows the case of specular reflectance of the docking interface, affecting the appearance of the LED marked by the arrow. The arrow in (c) indicates an LED-like reflection in the cover of the manipulator.

Thus, we decided to implement a simulation of the rover-camera-payload set-up, using the Gazebo simulator [63], to be able to conduct the first evaluations of the closed-loop control scheme without having access to the rover. The simulation will be described more in depth in chapter 7. The integration of the Gazebo simulator, together with all packages needed to ensure the correct functioning of the inter-process communication via ROS, into the newly released institute-internal package management system (Cissy) posed an additional, unforeseen and time consuming challenge.

5.2. System Concept

For the vision-based docking procedure, we consider a static scene. The docking target is placed within the reachable space, also referred to as work space, of the robotic arm. Thus, our focus lies solely on the tracking and servoing of the manipulator, excluding the rover base from our considerations. An overview on reference frames that will be frequently referred to over the following chapters can be found in Fig 5.3.
The overall pipeline can be subdivided into a tracking and a visual servoing part, each featuring further subtasks as can be seen in the general overview given in Fig. 5.4. The planned procedure is as follows:

The available data obtained by the forward kinematics of the manipulator provides us with a current a priori estimate of the relative pose of the flange relative to the manipulator base ${}^{B}\boldsymbol{T}_{F}$, which lets us define an initial region of interest (ROI) within the

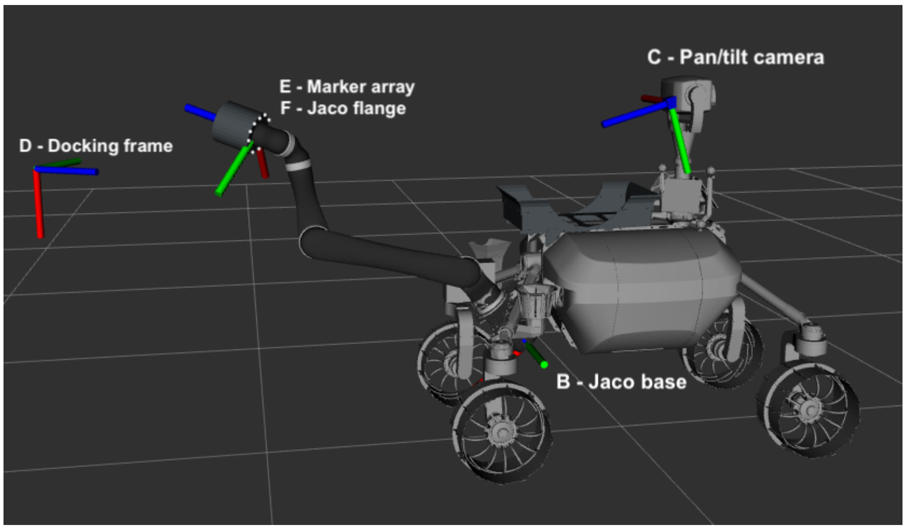


Figure 5.3.: Overview on important reference frames of the rover, frequently referred to in the following chapters. In absence of any kinematic error, the frame F of the manipulator's flange would coincide with that of the LED array E . The frame C depicted for the pan/tilt camera marks the left camera of the stereo setup. The docking goal frame D marks the desired position of E after successful docking. Not depicted is the object reference frame O, obtained by the AprilTag detection.

image, where the marker array is expected to be visible.

An initial detection step for the active marker array is performed on the first available image of the input stream, using the *iris filter detector* [5]. It finds the array by detecting circular marker point candidates, based on the calculation of a convergence score of the gradient for each pixel within the ROI. The convergence score considers both the direction and magnitude of the gradients, in order to find especially bright regions. In a subsequent step, an ellipse is fitted to the marker candidates to remove outliers. Output of this detection step is the relative transformation ${}^{C}\boldsymbol{T}_{E}$ between the camera frame and the reference frame of the detected LED array, and the list containing the image coordinates of the detected marker points.

Since the camera frame is part of the transformation tree, we can retrieve the relative transformation ${}^{F}\boldsymbol{T}_{E}$ between the flange, as given by the kinematics and vision respectively, by a simple concatenation of transformations [5]. This is then appended to the tree as a static transformation and represents the corrected flange frame pose estimate at time step t_0 [5]. By appending it as a static transformation, the dynamic change in error can be neglected as long as the pose difference between the detection and tracking step is small enough compared to the displacement of the end-effector in the horizontal plane. The retrieved list of image coordinates enables us to associate

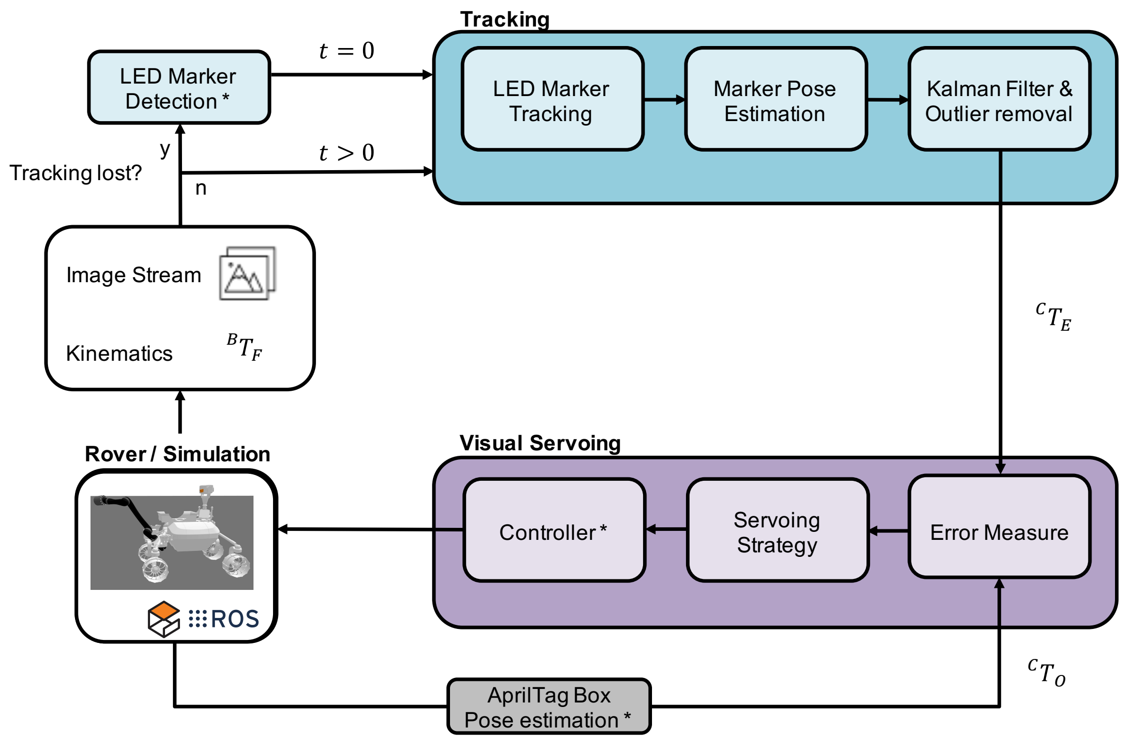


Figure 5.4.: Proposed system architecture. Elements marked with an asterisk are already provided. The detection step is only applied in the first image of the image stream, and to re-detect the marker array in the case that the tracking is lost.

the detected image points to their corresponding 3D model points. This is done by projecting the 3D model to the image plane and finding the closest detected marker point, based on their euclidean distance.

At the next time step t_1, i.e. with the next incoming image, the detection result is used as a starting point for the tracking of the LEDs. The image coordinates of the tracked points are then used to retrieve an updated estimate of the corrected flange pose ${}^{F}\boldsymbol{T}_{E}$. The subsequent application of an Error-State Extended Kalman filter (ES-EKF) reduces the noise and removes outliers in the 3D estimates.
The corrected flange frame will then be used together with the relative transform ${}^{C}\boldsymbol{T}_{O}$ between the left pan/tilt camera and the payload box, obtained from the AprilTag detector, to define an error measure for the visual servoing pipeline in the task space. Depending on the current docking phase, one of two visual servoing schemes will be used to drive the error towards zero: A position-based visual servo for the approach, and a hybrid-impedance based visual servo for the active docking phase, which will both supply the command input for the control of the manipulator.

In the following chapters, we will give a more in-depth description of the tracking pipeline and visual servoing strategies, respectively.

6. Tracking the Envicon Docking Interface

To improve the payload docking procedure through the usage of visual feedback, the robust tracking of both the end-effector and docking target is paramount. While the target payload box can be easily tracked using the provided AprilTags, tracking of the Envicon interface poses a more challenging task. For the tracking to be successful, our approach needs to meet several requirements. It has to be fast enough to continuously deliver updated pose estimates to correct for the dynamic error, show robustness to illumination changes and be reliable and stable. It further needs to ensure that pose outliers are filtered out to prevent supplying the visual servoing pipeline with incorrect pose estimates, as this could in turn lead the arm to move in a wrong direction.

To track the set of active markers within the image stream and provide a correct updated pose estimate of the docking interface at each time step, the tracking pipeline is thus subdivided into three consecutive segments:

1. tracking of the single LEDs within the image stream using our proposed Gradient Intersect Tracking method and retrieval of their 2D image coordinates and uncertainties,
2. estimation of the marker array's 3D pose, by solving the perspective-n-point (PnP) problem and propagating the image uncertainties, and
3. the application of an Error-State Kalman filter to filter out pose outliers and reduce the tracking noise introduced by the imprecision in subsequent PnP results obtained over the image stream.

6.1. Gradient Intersect Tracking

With our gradient intersect tracking (GIT) method, we present a simple and robust approach to keep track of the given LEDs. It does so by taking advantage of the manipulator's forward kinematics and the detection or tracking results of the last time step, using them as a prior to obtain an updated measure of the end-effector's position error. Based on this information we conduct a pose refinement step on the available monocular image data of the rover. By combining both knowledge about the kinematics and the vision measurements within our active marker tracking we aim at increasing the robustness of the algorithm to changes in the illumination conditions and visibility of the markers.

We start our description of the GIT method based on the assumption that all visible LEDs have been detected at t_0, yielding the error transform ${}^{F}T_{E}$ and that this transform is connected to the rover's transformation tree as static transformation. To predict the 3D movement of the marker array induced by the movement of the arm in between consecutive vision updates, we use a geometrical 3D model defining the positions of the LEDs relative to the reference frame E at the center of the active marker array. Thus, if the manipulator's end-effector moves, so does the frame E and our geometric model.

With the next incoming image at t_1, we project the 3D points ${}^{E}\boldsymbol{X}$ of the geometric model onto their image plane coordinates $\boldsymbol{x}_{img}$. This is done using the camera model as introduced in Sec. 4.1.2, Eq. 4.3

$$\boldsymbol{x}_{img} = \boldsymbol{K}\boldsymbol{P}\,{}^{E}\boldsymbol{X}, \tag{6.1}$$

such that $\boldsymbol{P}$ is the relative transformation ${}^{C}\boldsymbol{T}_{E}$ between the camera and the LED model frame. The image coordinates of the previously detected marker points are used as initial seed points for our tracking. The tracking then corresponds to a refinement step necessary to account for the dynamic property of the position error. Note that not all 12 marker points, but only those that were previously detected are being used as seed points. This is done to ensure no LED-like reflections are being falsely tracked, leading to a potentially distorted pose estimate in the next step.

For each seed point $\boldsymbol{p} = \begin{bmatrix} x_0 & y_0 \end{bmatrix}$ we define a fixed ROI of size $s = r \times c$ pixels, where r and c indicate the height and width of the ROI, respectively. We then continue by calculating the gradient field for the ROI. The image gradient is a measure of directional change of intensity in an image. The gradient vector $\boldsymbol{g}$ at point $\boldsymbol{x} = \begin{bmatrix} x & y \end{bmatrix}$ is defined by the partial derivative of its intensity $\boldsymbol{I}$ in the vertical and horizontal direction of the image [64]

$$\boldsymbol{g}(x,y) = \begin{bmatrix} \frac{\delta \boldsymbol{I}}{\delta x} & \frac{\delta \boldsymbol{I}}{\delta y} \end{bmatrix} \tag{6.2}$$

and always points in the direction of the largest intensity increase [64]. The vectors length corresponds to the magnitude of the gradient [64].

We then use a fixed number of m iterations to refine the seed point position. At each iteration, we start by defining N lines going through the current seed point coordinates. Along each line we then sample the gradient $\boldsymbol{g}_{max}$ with the largest magnitude within a radius r around the current seed point.

We initialize an empty array of size $s * rsf$, where rsf denotes the ROI scaling factor, an additional constant used to obtain sub pixel accuracy in the upcoming steps. This array serves as the *intersection image* $\boldsymbol{M}_{inter}(x,y)$. In this intersection image, we plot the lines defined by the previously retrieved vectors of maximum gradients $\boldsymbol{g}_{max}$. Plot in this case means, that we add a 1 to each element of the intersection image corresponding to a point on one of the lines $\boldsymbol{g}_i$ in $\boldsymbol{g}_{max}$. This approach has been chosen to reduce the computational time needed for the calculation of the intersection points, compared to using other geometrical methods. Every intersection point of two or more lines is thus marked by entries $\boldsymbol{M}_{inter}(x,y) \geq 2$. We then proceed to set all entries smaller than two equal to zero and are left with an image of weighted intersection points.

Ideally, all gradient lines in $\boldsymbol{M}_{inter}$ would meet in one single point, marking the LED center. This is the case when the displacement of the end-effector between the current and the last image is small and the marker point corresponding to the seed point is visible as a perfectly bright spot in front of an otherwise dark background. However, in reality the image of the LED might have a more elliptical shape, or specular reflections at the edge of the LED might give it a less circular appearance. Both cases are depicted in Fig. 6.1.

Due to this, we are normally presented with more than one intersection point in $\boldsymbol{M}_{inter}$. By calculating the image moments we can retrieve the centroid $\boldsymbol{x}' = \begin{bmatrix} x' & y' \end{bmatrix}$ of the intersection image together with its covariance Σ'. The moment of order $(i+j)$ for $i, j = 0, 1, 2...$ of an image with pixel intensities $\boldsymbol{I}(x,y)$ is defined by [65]

$$M_{ij} = \sum_x \sum_y x^i y^j \boldsymbol{I}(x,y). \tag{6.3}$$

The centroid of the image is then given by $\boldsymbol{x}' = \begin{bmatrix} \frac{M_{10}}{M_{00}} & \frac{M_{01}}{M_{00}} \end{bmatrix}$ [65]. This result is used to update the seed point and the refinement is continued until the maximum number of iterations is reached. We have found that a number of $m = 5$ iterations yields the best results, while staying computationally efficient.

In the last iteration the covariance matrix of the refined seed point is calculated using the second-order central moments [65]

$$\mu'_{20} = \frac{M_{20}}{M_{00}} - x'^2, \tag{6.4}$$

$$\mu'_{02} = \frac{M_{02}}{M_{00}} - y'^2, \tag{6.5}$$

$$\mu'_{11} = \frac{M_{11}}{M_{00}} - x'y', \tag{6.6}$$

$$\tag{6.7}$$

describing the uncertainty in the center point estimate. The covariance matrix of the tracked point is then given by

$$\Sigma' = \begin{bmatrix} \mu'_{20} & \mu'_{11} \\ \mu'_{11} & \mu'_{02} \end{bmatrix} \tag{6.8}$$

and returned together with the updated center coordinates of the marker point. A summary of the algorithm is given in appendix A.1.

It should be noted that we have chosen not to update the positions of the ROIs within

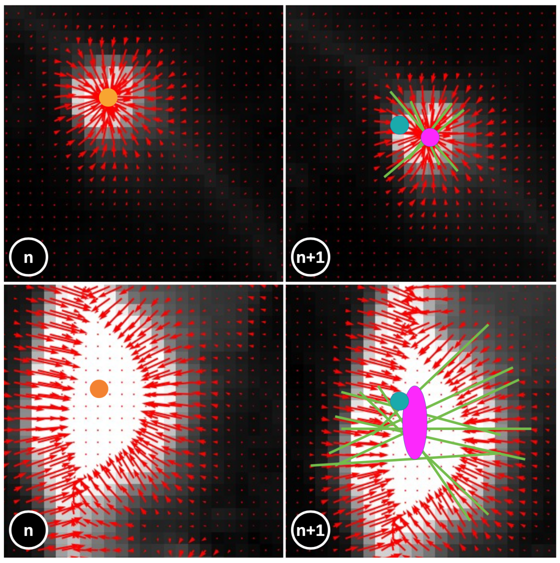


Figure 6.1.: Two different cases of tracked LEDs. The top row shows an ideal case with circular marker appearance, while the lower shows an elongated appearance, due to a more tilted end-effector pose and different illumination conditions. The left column illustrates the detection result at time step n, with the orange point marking the detected center x. The right column shows the tracking update in the next time step. The turquoise point is the initial seed point while the pink ellipse depicts the covariance Σ' of the updated seed point x' at its center after $m = 5$ iterations. The green lines correspond to the directions of the maximum gradients in the last iteration step.

the image between iterations, as we found out that doing so increased the wandering-off of the centroids in the presence of strong specular reflections near the border of the marker points.

6.2. Marker Pose Estimation

To estimate the position and orientation of a calibrated camera from known 3D-to-2D point correspondences between a 3D model and their projections in the image plane, the so called PnP problem [66] has been proposed by Fischler, in the early 1980s. It is a fundamental part of many robotics applications, augmented reality, and simultaneous localization and mapping. Due to its importance, a vast number of algorithms has been proposed to date, to solve this problem.

With our knowledge about the camera's intrinsic parameters, the 2D image coordinates of the tracked marker points, and the corresponding 3D geometrical model of our marker array, we can thus proceed to estimate the pose of the camera relative to our marker array by solving the PnP problem. The pose is thereby found by solving the equation of the camera model 4.1.2 for the matrix of extrinsic parameters $\boldsymbol{P} = [\mathbf{R}|\mathbf{t}]$, where $\mathbf{R}$ is the rotation and $\mathbf{t}$ the translation of the camera relative to the world coordinates of the marker array.

Since the pose of the camera has a total of six DOF and each 2D-to-3D correspondence determines two equations, generally a minimum of $n = 3$ points is needed to solve for the whole pose. This is called the P3P problem and yields up to four geometrically feasible solutions [67]. A fourth point-to-point correspondence can be used to find a unique solution [67]. Up do date, many PnP solutions for $n \geq 4$ points [67–69] have been presented.

6.2.1. Uncertainty Propagation using MLPnP

We decided to use maximum likelihood perspective-n-point (MLPnP) [22], which formulates a maximum likelihood solution to the PnP problem and extends the previously mentioned PnP methods by taking into account the observation uncertainty of the image points, given by Eq. 6.8. The image uncertainties are then propagated throughout the algorithm, consisting of a first linear maximum likelihood estimation followed by an iterative Gauß-Newton refinement step, to obtain the corresponding uncertainties along all DOF of the returned 3D pose estimate.

To be applicable to any arbitrary central camera model, MLPnP works with 3D bearing vectors instead of 2D image points [22]. A bearing vector is a unit vector describing the direction of a point relative to the camera frame. More precisely, the variances and image observations need to be propagated to the nullspace of their bearing vectors to avoid singular covariance matrices [22], as we will see in the following. Recalling Sec. 4.3.4, we note that this step is necessary to assure the convergence of the subsequent Gauß-Newton refinement of the initial maximum likelihood pose estimate.

The propagation is done as described in [22]:

First, a retrieved image point $\boldsymbol{x}'$ is forward projected to its homogeneous coordinate $\boldsymbol{x}$ in the camera frame using the camera's matrix of intrinsic parameters $\boldsymbol{K}$

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \boldsymbol{K}^{-1}\boldsymbol{x}'. \tag{6.9}$$

Using the Jacobian $\boldsymbol{J}_{K^{-1}}$ of the forward projection, the covariance matrix of the image point, indicating the uncertainty of its retrieved center coordinates, is propagated by [22]

$$\boldsymbol{\Sigma}_{\boldsymbol{x}} = \boldsymbol{J}_{K^{-1}}\boldsymbol{\Sigma}'\boldsymbol{J}_{K^{-1}}^{T} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & 0 \\ \sigma_{yx} & \sigma_y^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{6.10}$$

with [22]

$$\boldsymbol{J}_{K^{-1}} = \begin{bmatrix} \frac{\delta \boldsymbol{K}_{x'}^{-1}}{\delta x'} & \frac{\delta \boldsymbol{K}_{x'}^{-1}}{\delta y'} \\ \frac{\delta \boldsymbol{K}_{y'}^{-1}}{\delta x'} & \frac{\delta \boldsymbol{K}_{y'}^{-1}}{\delta y'} \\ 0 & 0 \end{bmatrix}. \tag{6.11}$$

In a further step, a spherical normalization yields the bearing vector and its covariance [22]

$$\boldsymbol{v} = \frac{\boldsymbol{x}}{||\boldsymbol{x}||} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{\Sigma}_{\boldsymbol{v}} = \begin{bmatrix} \sigma_{v_x}^2 & \sigma_{v_{xy}} & \sigma_{v_{xz}} \\ \sigma_{v_{yx}} & \sigma_{v_y}^2 & \sigma_{v_{yz}} \\ \sigma_{v_{zx}} & \sigma_{v_{zy}} & \sigma_{v_z}^2 \end{bmatrix}, \tag{6.12}$$

where $\boldsymbol{\Sigma}_{\boldsymbol{v}}$ is obtained by [70]

$$\boldsymbol{\Sigma}_{\boldsymbol{v}} = \boldsymbol{J}\boldsymbol{\Sigma}_{\boldsymbol{x}}\boldsymbol{J}^{T}, \tag{6.13}$$

with

$$\boldsymbol{J} = \frac{1}{||\boldsymbol{x}||}(\boldsymbol{I}_3 - \boldsymbol{v}\boldsymbol{v}^T). \tag{6.14}$$

Note that at this point the covariance matrix is still singular.

Lastly, the bearing vector is projected on to its tangent space, spanned by the axes $\boldsymbol{r}$ and $\boldsymbol{s}$ given by [22]

$$\begin{bmatrix} \boldsymbol{r} & \boldsymbol{s} \end{bmatrix} = \begin{bmatrix} r1 & s1 \\ r2 & s2 \\ r2 & s3 \end{bmatrix} = \boldsymbol{J}_{v_r}(\boldsymbol{v}), \tag{6.15}$$

where $\boldsymbol{J}_{v_r}(\boldsymbol{v})$ corresponds to the matrix represented by the eigenvectors of the two zero eigenvalues obtained by the singular value decomposition of $\boldsymbol{v}^T$.

The projection yields its vector of residuals $\boldsymbol{v}_r$ given by [22]

$$\boldsymbol{v}_r = \begin{bmatrix} \delta r \\ \delta s \end{bmatrix} = \boldsymbol{J}_{v_r}^T(\boldsymbol{v})\boldsymbol{v}, \tag{6.16}$$

with covariance

$$\boldsymbol{\Sigma}_{\boldsymbol{v}_r} = \boldsymbol{J}_{v_r}^T\boldsymbol{\Sigma}_{\boldsymbol{v}}\boldsymbol{J}_{v_r} = \begin{bmatrix} \sigma_{v_{rx}}^2 & \sigma_{v_{rxy}} \\ \sigma_{v_{ryx}} & \sigma_{v_{ry}}^2 \end{bmatrix}. \tag{6.17}$$

$\boldsymbol{v}_r$ and $\boldsymbol{\Sigma}_{v_r}$ are then used by MLPnP for the calculation of a linear maximum likelihood estimate (see Sec. 4.3.3) of the extrinsic parameters, by minimizing $\boldsymbol{v}_r$ in the tangent space [22]. It has to be noted that in this step the extrinsic parameters to be recovered are represented by a rotation matrix and a translation, presenting us with 12 unknowns. Thus, as we obtain two residuals from each observed world point $\boldsymbol{x}_w$

$$\begin{bmatrix} \delta r \\ \delta s \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}^T \\ \boldsymbol{s}^T \end{bmatrix} (\boldsymbol{R}\boldsymbol{x}_w + \boldsymbol{t}) = 0, \tag{6.18}$$

a minimum of $N = 6$ points need to be visible within the image to be able to apply MLPnP and obtain a full pose estimate.

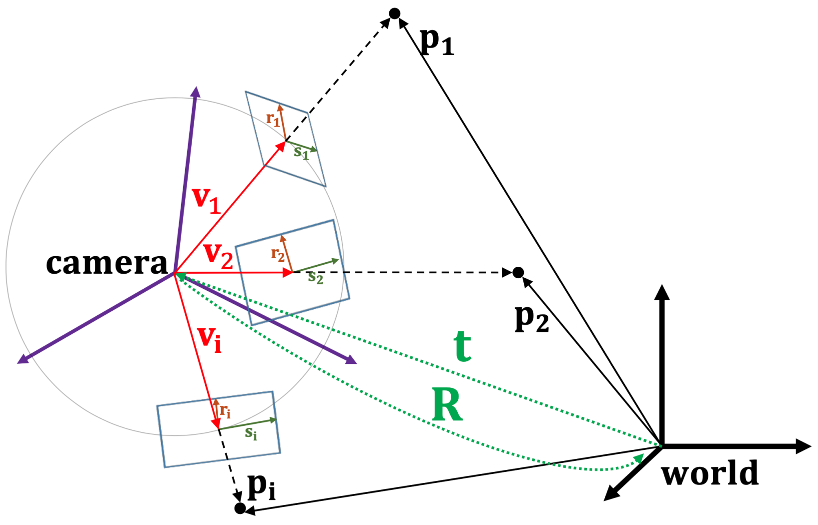


Figure 6.2.: The normalized bearing vectors $\mathbf{v}_i$ indicate the direction to the object points $\boldsymbol{p}_i$ observed by a camera. For each bearing vector its null space vectors $\boldsymbol{r}_i$ and $\boldsymbol{s}_i$ span a plane which is tangential to the unit sphere. Taken from [22].

For the linear estimation of the camera pose the residuals of all N points are stacked in a design matrix $\boldsymbol{A}$, while their respective covariance matrices are combined to form the matrix of the stochastic model $\boldsymbol{P}$ given by [22]

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{\Sigma}_{v_r}{}^{-1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \boldsymbol{\Sigma}_{v_r}{}^{-1} \end{bmatrix}. \tag{6.19}$$

This is used to construct a linear system of equations and solving it for the elements of the camera's extrinsics $\boldsymbol{u} = [r_{11}\ r_{12}\ r_{13}\ r_{21}\ r_{22}\ r_{23}\ r_{31}\ r_{32}\ r_{33}\ t_1\ t_2\ t_3]^T$ [22]

$$\boldsymbol{A}^T\boldsymbol{P}\boldsymbol{A}\boldsymbol{u} = \boldsymbol{N}\boldsymbol{u} = \boldsymbol{0}. \tag{6.20}$$

In a last step, the linear estimate is refined by Gauß-Newton optimization, as introduced in Sec. 4.3.4. This refinement yields the final pose estimate $\boldsymbol{x}_{tr}$ of the LED array in a minimal form, with the rotation following the axis/angle convention

$$\boldsymbol{x}_{tr} = \begin{bmatrix} x \\ y \\ z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, \tag{6.21}$$

and the vector of standard deviations of the pose $\boldsymbol{\sigma_{tr}}$, out of which we form the covariance matrix $\boldsymbol{\Sigma}_{tr}$ of the pose

$$\boldsymbol{\Sigma}_{tr} = \begin{bmatrix} \sigma^2_{tr_x} & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{tr_y} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{tr_z} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{tr_{\omega_x}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{tr_{\omega_y}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^2_{tr_{\omega_z}} \end{bmatrix}. \tag{6.22}$$

Subsequently, $\boldsymbol{x}_{rt}$ is either used to directly update ${}^F\boldsymbol{T}_E$, or as a measurement input for the ES-EKF presented in Sec.6.3.3, together with its estimated uncertainty $\boldsymbol{\Sigma}_{tr}$.

As we have found that MLPnP shows a high sensibility to point outliers, we prepend it with a random sample consensus (RANSAC) [66] step. RANSAC iteratively solves the PnP problem to find the inliers of the data points that best fit the 2D-to-3D correspondences and lead to the smallest reprojection error, while neglecting the outliers. We used the PnP-RANSAC implementation included in the OpenCV library [71]. This implementation offers different PnP methods to choose from for the final pose estimation step, which is based on all previously found inliers. We chose the iterative solver, which utilizes the Levenberg-Marquard method, presented in Sec. 4.3.4. The decision is based on our finding that this method is the most robust one when confronted with our planar marker compared to the other implemented methods.

If $N \geq 6$ inliers are found, MLPnP is carried out as described above to yield a refined solution and propagate the uncertainties. Otherwise, for $4 \leq N \leq 5$ inliers, the result of the PnP-RANSAC algorithm is used directly. To be used in the ES-EKF presented in Sec. 6.3.3, these results are assigned a constant covariance matrix, which was manually tuned to $\mathbf{\Sigma}_{tr} = \mathrm{diag}(0.005) \in \mathbb{R}^{6\times 6}$.

Lastly, in case of $N \leq 3$ inliers, we assume that the tracking has been lost and a new detection is triggered. The covariance of the detection outputs was tuned to $\mathbf{\Sigma}_{tr} = \mathrm{diag}(0.05) \in \mathbb{R}^{6\times 6}$. The higher uncertainty in the detection, compared to the tracking result, stems from the fact that the detector does not explicitly aim to find the centers of the marker points, thus yielding a less accurate PnP result.

6.2.2. Evaluation - Tracking and Pose Estimation

For an off-line evaluation of the performance of our GIT tracker and pose estimation, we recorded several ROS Bags containing the kinematic and vision data obtained along a predefined end-effector trajectory and in different lighting conditions. The trajectory was chosen to include a wide range of different end effector poses, in order to test the validity of the tracking when confronted with easily trackable and more challenging views of the marker array. The former correspond for example to poses where the marker array is oriented nearly perpendicular to the optical axis of the camera. The latter are given, e.g., by strongly tilted or partially occluded views. To obtain information about the accuracy of the vision-corrected position estimates we recorded the ground-truth information of the end-effector position using a Vicon® tracking system. The variables of our GIT tracker have been determined empirically and can be found in Tab. 7.3.

ROI size *s* [pixel]	**Radius *r* [pixel]**	**Number of lines *N***	**ROI Scaling Factor *rsf***
20×20	8	20	8

Table 6.1.: Chosen values for the variables of the gradient intersect tracker.

We start by evaluating the correction of the end-effector position. It has to be noted, that system overload during the recording process led to repeated lag in the ground-truth signal, which resulted in strong error peaks where the ground-truth lagged behind the kinematic and vision measurements. Due to this, the data only allows for a qualitative, and not a quantitative comparison of the position errors. All values referred to in the following paragraphs should thus be treated as mere approximations to the real underlying values.

The qualitative comparison of the normed position errors, presented in Fig. 6.3, shows that the tracking reduces the error of the measured end-effector position considerably over the whole trajectory. While the kinematic measurements exhibit errors as high as 28 mm, the maximum tracking error for the given trajectory amounts to approximately 10 mm in sections where new vision updates were available. Even in cases when no new vision updates could be acquired for an extended amount of time, e.g., due to occlusion of the marker array, the tracking shows good robustness and still outperforms the results of the kinematic measurements by up to 77 %. This is due to the inclusion of the tracking frame as part of the transformation tree of the rover.

At approximately 60 s and 145 s the plot shows spikes in the tracking error, where the kinematic error stays constant. Unlike the large error peaks introduced by the ground-truth lag, which affects both error signals equally, these tracking spikes indicate outliers in 3D position estimates of the vision pipeline due to the false detection of led-like reflections in the image. Since the tracker can not distinguish between *true* and *false* leds, an additional outlier removal step, applied to the 3D estimates, is necessary to prevent wrong position estimates from being used for the visual servoing of the arm. This will be considered in section 6.3.4

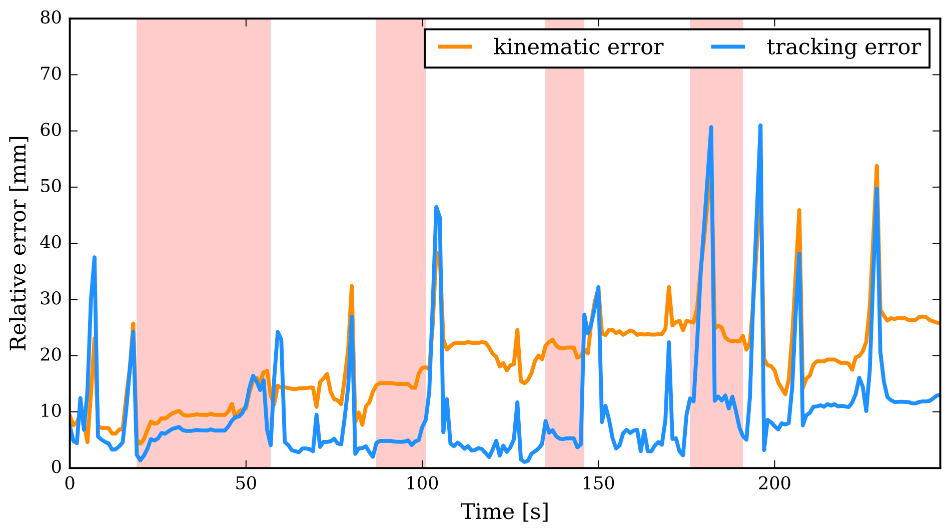


Figure 6.3.: Comparison of the normed position error of the end-effector as given by the kinematic measurements alone (orange) and our vision correction (blue). Both are given relative to the ground truth obtained by the Vicon® tracking. The red background indicates periods in which no tracking update could be acquired, e.g., due to occlusion of the end-effector.

Fig. 6.4 further shows the absolute values of the kinematic and tracking error of the same data-set, broken down to the three end-effector axes. It is apparent, that the tracking reduces the position error along the X and Y directions, where it remains below the threshold of 10 mm necessary to allow for the successful docking.

A comparison of the normed kinematic and tracking position errors as function of the relative horizontal arm stretch is shown in Fig. 6.5. A stretch of zero indicates the manipulator being fully perpendicular to the ground, while a value of one means the manipulator is fully stretched out in the horizontal plane, parallel to the ground. The

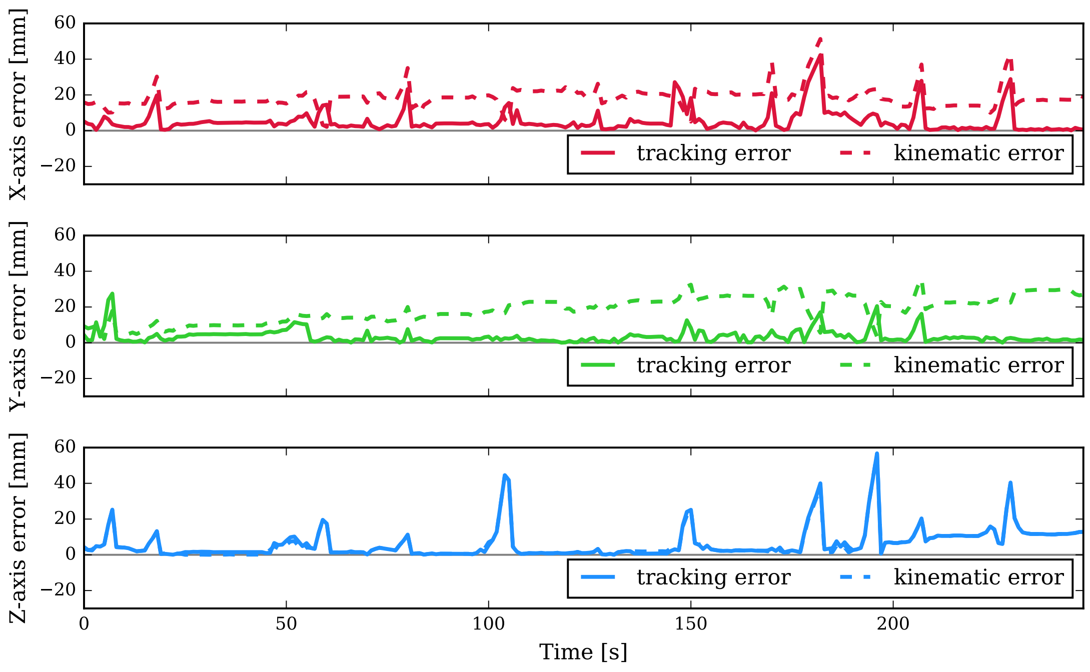


Figure 6.4.: Comparison of the kinematic and tracking error along the axis of the end-effector coordinate system. The Z-axis corresponds to the center axis of the docking cylinder.

dots indicate single measurements while the lines showing the qualitative progress of the measurements are fitted using LOWESS regression. It can be seen that, while the kinematic error increases significantly over the course of the stretch, the tracking error shows only a light increase for medium values of the arm stretch. This is likely caused by more tilted views of the end-effector in medium-stretched configurations of our chosen trajectory. The tracking successfully improves the position estimate, especially for those arm poses most affected by the effects of gravity on the fully stretched manipulator.

The evaluation of the quality of the visual rotation estimates obtained by the tracking pipeline showed that our chosen PnP method did not perform as intended. Tab. 6.2 shows a comparison of the end-effector rotation errors compared to the Vicon® ground truth, as given by the kinematic measurements and vision estimates, respectively. Especially the errors around the X and Y axes are of importance when considering the execution of the docking procedure for our end-effector, while the error around the Z-axis is negligible due to the rotational symmetry of the docking interface. While the rotation obtained by the kinematics displays mean errors in the range of 1.67° to 3.84°, those recovered by MLPnP are significantly higher. This can have a number of reasons. While the MLPnP algorithms needs a minimum of six point-to-point correspondences to return a result, the accuracy of the result strongly depends on the number of available

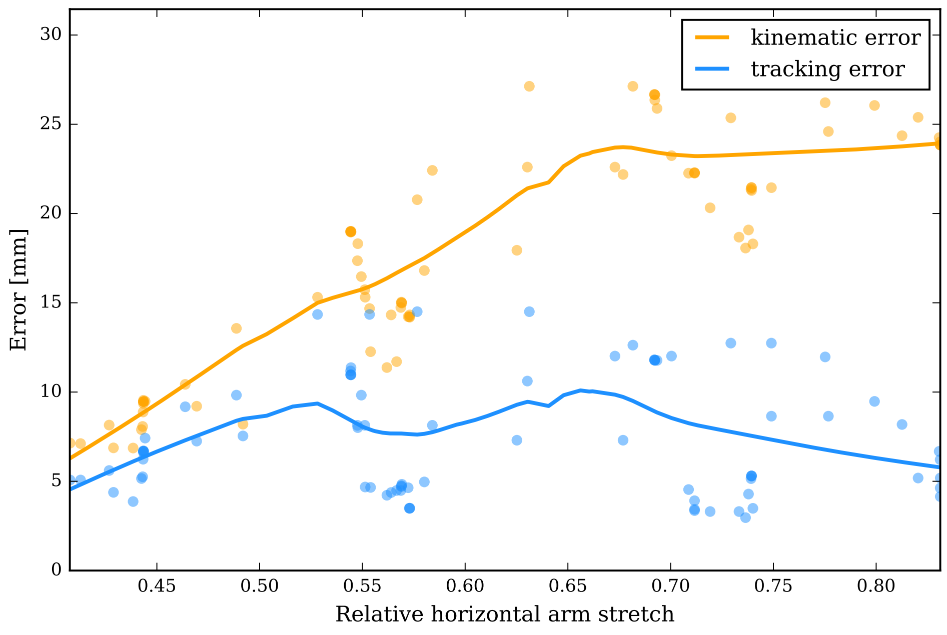


Figure 6.5.: The kinematic and tracking error depending on the horizontal stretch of the manipulator. Dots indicate the measured normed error, lines show the qualitative, LOWESS filtered error development. A horizontal stretch of zero corresponds to the arm being oriented perpendicularly to the ground, while a value of one indicates the manipulator being fully stretched out, parallel to the ground.

2D-to-3D point pairs. The more correspondences are found, the more accurate we can expect the result to be [22]. However, our set-up offers only a maximum of 12 LEDs to be tracked, of which rarely more than eight are visible within the image at the same time, due to partial occlusion of the marker array by proximal links of the manipulator. The planarity of our marker is a further factor that reduces the accuracy of the recovered pose [22]. Additionally, the quality of the results depends on the accuracy of the retrieved image point coordinates [22]. All these aspects affect the recovery of the correct rotation estimates.

Considering the low rotation error of our kinematic measurements, we have decided to limit ourselves to the correction of the position error of the end-effector. This simplification is applicable, since the main source of error in our kinematics is given by the translational component of the flange transform. Further, as we will see in section 7.3.2, the impedance based visual servoing approach planned for the docking phase tolerates a certain rotational deviation, while being sensitive to position errors, thus necessitating the position correction.

	$\Delta\omega_x$	$\Delta\omega_y$	$\Delta\omega_z$
	[°]	[°]	[°]
Kinematic	1.67 ± 2.01	3.84 ± 1.24	1.78 ± 1.62
MLPnP	9.75 ± 17.08	12.61 ± 22.82	1.27 ± 2.01

Table 6.2.: Comparison of the resulting mean and standard deviation of the rotation error $\Delta\omega$ for the kinematic measurements and MLPnP rotation estimates of the end-effector frame. Both are given relative to the ground truth, obtained by the Vicon® tracking.

6.3. Smoothing and Outlier Rejection with the Kalman Filter

The pose estimation pipeline described thus far supplies us with a good correction of the position estimate for the end-effector at each time step. However, fluctuations in the accuracy of the image point measurements, e.g., due to reflections close to the tracked LEDs, lead to jitter in the time-dependent position estimates. Further, whenever we lose tracking and need to re-detect the marker, LED-like reflections may lead to a false positive detection result, away from the actual end-effector position, which would result in the generation of an unwanted control signal if used within our visual servoing loop.

This motivates the use of a filter, paired with an additional outlier rejection, to obtain a smoother tracking result over time, which is less afflicted by jumps due to faulty detection results. We chose to implement an Error-State Kalman filter for this purpose.

6.3.1. Kalman Filter

The Kalman filter is a recursive estimator that attempts to predict state $\boldsymbol{x} \in \mathbb{R}^n$ of a discrete-time dynamic system at time step k by combining the knowledge about the state at step $k-1$ and incoming sensor measurements, in the presence of noise. In the case of a linear dynamic system the Kalman filter is an optimal mean square estimator.

The state evolution is defined by the process model [72, p. 539]:

$$\boldsymbol{x}_k = \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{G}\boldsymbol{u}_{k-1} + \boldsymbol{v}_k, \tag{6.23}$$

where $\boldsymbol{F} \in \mathbb{R}^{n\times n}$ is the state transition matrix applied to the previous state $\boldsymbol{x}_{k-1}$, describing the system dynamics, $\boldsymbol{G} \in \mathbb{R}^{n\times m}$ is the control-input matrix, relating the control vector $\boldsymbol{u} \in \mathbb{R}^m$ to the system. The process noise $\boldsymbol{v} \in \mathbb{R}^n$ is a Gaussian random variable, $\boldsymbol{v}_k \sim \mathcal{N}(0, \boldsymbol{Q})$, with zero mean and covariance $\boldsymbol{Q}$, depicting unmodeled disturbances and errors in the matrices $\boldsymbol{F}$ and $\boldsymbol{G}$.

The relationship between a measurement and the state at the current time step k

is given by the measurement model [72, p. 539]:

$$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_{k-1} + \boldsymbol{w}_k, \tag{6.24}$$

where $\boldsymbol{z}_k \in \mathbb{R}^p$ is the measurement vector, $\boldsymbol{H} \in \mathbb{R}^{p \times n}$ is the observation model describing the mapping between system state and observed outputs. The measurement noise $\boldsymbol{w} \in \mathbb{R}^p$ is an other random variable, $\boldsymbol{w}_k \sim \mathcal{N}(0, \boldsymbol{S})$, taking into account the imperfections of the observation model $\boldsymbol{H}$. While the covariance matrices $\boldsymbol{Q}$ and $\boldsymbol{S}$ are supposed to reflect the noise statistics, their true statistics are often unknown or non-Gaussian [73]. Therefore, both matrices can be regarded as tuning values that can be manually adjusted to yield the desired behavior [73]. If $\boldsymbol{Q}$ is set high compared to $\boldsymbol{S}$, the process model is considered to be less reliable than the measurements and, thus, the measurement inputs are followed more closely. To obtain a good smoothing behavior, the values of $\boldsymbol{Q}$ have to be set small enough in relation to $\boldsymbol{S}$ to ignore spikes in the incoming measurement, but not too small as that would mean that the filter becomes slow to respond to changes in the measurements. Finding a good balance between these two factors can present a challenging task.

The Kalman filter is initialized with a first guess of the state estimate $\hat{x}_0$ and error covariance $\boldsymbol{P}_0$. After this, filtering consists of two steps, which are applied recursively: prediction and correction, as depicted in Fig. 6.6.

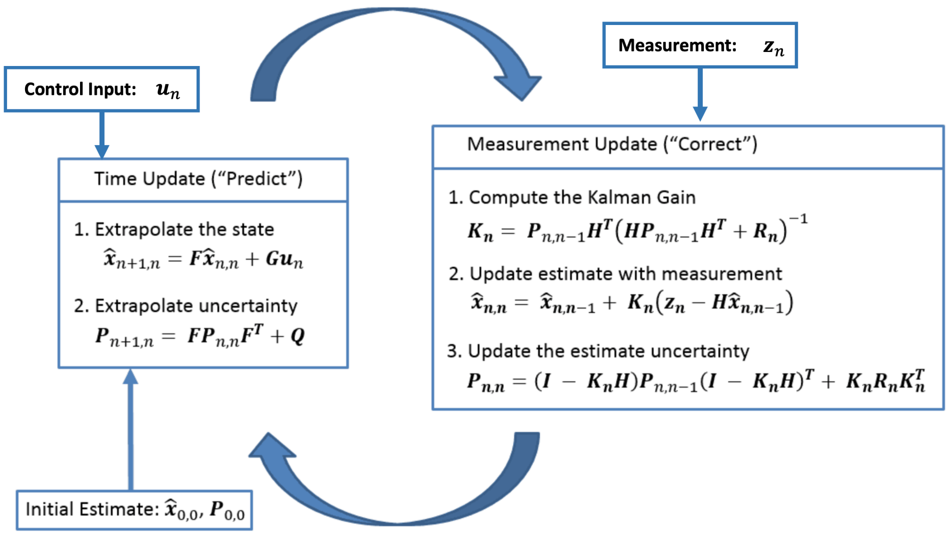

Control Input: $\boldsymbol{u}_n$
Measurement: $\boldsymbol{z}_n$
Time Update ("Predict")
1. Extrapolate the state
$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_n$
2. Extrapolate uncertainty
$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q}$
Measurement Update ("Correct")
1. Compute the Kalman Gain
$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T\left(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n\right)^{-1}$
2. Update estimate with measurement
$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n\left(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1}\right)$
3. Update the estimate uncertainty
$\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\boldsymbol{P}_{n,n-1}(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$
Initial Estimate: $\hat{\boldsymbol{x}}_{0,0}, \boldsymbol{P}_{0,0}$

Figure 6.6.: The concept of a Kalman filter. Adapted from [74]

In the following, the notation $\hat{\boldsymbol{a}}_{k|k-1}$ stands for the *estimate* of a variable $\boldsymbol{a}$ at time k using observations up to and including time step $k-1$.

Prediction

In the prediction step, the estimated state $\hat{\boldsymbol{x}}_{k|k-1}$ and the covariance of the state error $\boldsymbol{P}$ at time k are calculated using the last state and the inputs applied to it with [72, p. 539]:

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{G}\boldsymbol{u}_k \tag{6.25}$$

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}^T + \boldsymbol{Q} \tag{6.26}$$

Correction

When a new measurement is received, a correction step is performed [72, p. 539]:

$$\bar{\boldsymbol{y}}_k = \boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_k \tag{6.27}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^T + \boldsymbol{S})^{-1} \tag{6.28}$$

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\bar{\boldsymbol{y}}_k \tag{6.29}$$

$$\boldsymbol{P}_{k|k} = \boldsymbol{P}_{k|k-1} - \boldsymbol{K}_k\boldsymbol{H}\boldsymbol{P}_{k|k-1} \tag{6.30}$$

The correction step starts with the calculation of the innovation $\bar{\boldsymbol{y}}_k \in \mathbb{R}^p$, reflecting the difference between the true measurement and its estimated value. $\boldsymbol{K}_k \in \mathbb{R}^{n\times n}$ is the Kalman gain and $\hat{\boldsymbol{x}}_{k|k}$ and $\boldsymbol{P}_{k|k}$ are the corrected state estimate and error covariance.

6.3.2. Extended Kalman Filter

To apply the standard Kalman filter, both the state transition and observation models need to be linear functions. The Extended Kalman filter (EKF) filter provides a tool to apply Kalman filtering to nonlinear dynamic systems, for which the process model and measurement model are given by the nonlinear functions [75]:

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k) + \boldsymbol{v}_k, \tag{6.31}$$

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{w}_k, \tag{6.32}$$

containing, e.g., constraint quantities such as rotations in 3D.
It does so by using the first derivative of their Taylor series, linearizing the process model around the latest estimate of the state, while linearizing the measurement model around the prediction of the state based on the motion model. This yields the following Jacobians [75]:

$$\boldsymbol{F}_k = \left.\frac{\delta \boldsymbol{f}}{\delta \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_k} \tag{6.33}$$

$$\boldsymbol{H}_k = \left.\frac{\delta \boldsymbol{h}}{\delta \boldsymbol{x}}\right|_{\boldsymbol{x}_{k|k-1}} \tag{6.34}$$

The system is now linear in the state space and the standard Kalman equations of the prediction and correction step can be applied by replacing the state transition and observation model by their Jacobians. The only two exceptions are Eqs. 6.25 and 6.28, which instead use the nonlinear models for the propagation of the state estimate and the computation of the innovation [75]:

$$\hat{\boldsymbol{x}}_{k|k-1} = \mathbf{f}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_k) \tag{6.35}$$

$$\bar{\boldsymbol{y}}_k = \boldsymbol{z}_k - \mathbf{h}(\hat{\boldsymbol{x}}_k) \tag{6.36}$$

$$\tag{6.37}$$

6.3.3. Error-State Extended Kalman Filter

Even though only the translational component of the end-effector pose is corrected by our vision pipeline, the kinematic measurements ${}^C\boldsymbol{T}_F = \begin{bmatrix}\boldsymbol{q}_F & {}^C\boldsymbol{t}_F\end{bmatrix}$ still supply us with the corresponding end-effector orientation, which we wish to filter too. For the orientation to be used as part of the state in the EKF introduced above, the rotational component would need to be given in a minimal representation, since the EKF does not enforce the constraints of over parameterized rotation representations [76]. However, every minimal representation presents singularities, which we wish to avoid.

The ES-EKF, also referred to as multiplicative Kalman filter, avoids this problem by representing the global orientation with a quaternion, while employing a tangent space representation of the rotation for the error as part of the state [77]. More specifically, the ES-EKF performs an unconstrained estimation of the tangent space error $\boldsymbol{\omega}$, while using a quaternion $\boldsymbol{q}_{ref}$ as a reference around which the error is defined [77]. The nominal orientation $\boldsymbol{q}$ is then given by the product [77]

$$\boldsymbol{q} = \delta\boldsymbol{q}(\boldsymbol{\omega}) \otimes \boldsymbol{q}_{ref}, \tag{6.38}$$

where $\delta\boldsymbol{q}(\boldsymbol{\omega})$ is the unit quaternion representing the rotation from $\boldsymbol{q}_{ref}$ to the nominal orientation.

Modeling the State

We treat the vision pipeline as a black box "sensor" and its result as a regular sensor measurement. Thus, two measurements have to be considered: the current pose from the kinematics ${}^C\boldsymbol{T}_F = \begin{bmatrix}\boldsymbol{q}_F & {}^C\boldsymbol{t}_F\end{bmatrix}$ and the position ${}^C\boldsymbol{t}_E$ received by our vision pipeline. Additionally, MLPnP supplies us with the covariance $\boldsymbol{\Sigma}_{tr}$ of the visual estimate.

To model the state vector $\boldsymbol{x}$ we first consider the predicted end-effector pose, as given by the manipulator kinematics. The first six elements of our state vector thus contain the minimal representation of the rotational bias between two kinematic updates, given as axis/angle vector $\boldsymbol{\omega}_F \in \mathbb{R}^3$ and the predicted translation ${}^C\boldsymbol{t}_F \in \mathbb{R}^3$ of the flange. In addition, we keep track of the reference quaternion of the flange frame $\boldsymbol{q}_{mem}$ which is not part of the state, but used to obtain update the nominal rotation as given in Eq. 6.38.

To model the dynamic relationship between the kinematic error and the stretch of the arm, the next element considers the translation bias between kinematic prediction and vision correction ${}^{F}\boldsymbol{t}_{E} \in \mathbb{R}^3$. Lastly, we keep track of the velocity ${}^{C}\boldsymbol{v} \in \mathbb{R}^3$ of the flange. The full state is thus given by:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\omega}_F \\ {}^{C}\boldsymbol{t}_F \\ {}^{F}\boldsymbol{t}_E \\ {}^{C}\boldsymbol{v} \end{bmatrix} \in \mathbb{R}^{12}. \tag{6.39}$$

Note that in the following $\boldsymbol{R}(\boldsymbol{q})$ will imply the quaternion to rotation matrix conversion. The conversion for the used rotation conventions can be revisited in Sec. 4.2.2.

The initialization of the filter is set with the first incoming kinematic measurement

$$\boldsymbol{x}_0 = \begin{bmatrix} \boldsymbol{0}_3 \\ {}^{C}\boldsymbol{t}_{F_0} \\ \boldsymbol{0}_3 \\ \boldsymbol{0}_3 \end{bmatrix}, \tag{6.40}$$

$$\boldsymbol{P}_0 = \boldsymbol{I} \in \mathbb{R}^{12\times 12}, \tag{6.41}$$

$$\boldsymbol{q}_{mem} = \boldsymbol{q}_{F_0}, \tag{6.42}$$

where $\boldsymbol{0}_3 \in \mathbb{R}^3$.

Prediction

For the prediction step, we assume the velocity of the end-effector movement to be constant during the sampling interval and apply a constant velocity motion model [78]. The state propagation model is then given by

$$\hat{\boldsymbol{\omega}}_{F_k} = \hat{\boldsymbol{\omega}}_{F_{k-1}}, \tag{6.43a}$$

$${}^{C}\hat{\boldsymbol{t}}_{F_k} = {}^{C}\hat{\boldsymbol{t}}_{F_{k-1}} + {}^{C}\hat{\boldsymbol{v}}_{k-1}\,\Delta t, \tag{6.43b}$$

$${}^{F}\hat{\boldsymbol{t}}_{E_k} = {}^{F}\hat{\boldsymbol{t}}_{E_{k-1}}, \tag{6.43c}$$

$${}^{C}\hat{\boldsymbol{v}}_k = {}^{C}\hat{\boldsymbol{v}}_{k-1}, \tag{6.43d}$$

where Δt is the operating rate of the Kalman filter.

The state covariance is propagated as given in Eq. 6.26, with the state transition matrix of Eqs. 6.43 being given by

$$\boldsymbol{F}_k = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{I}\Delta t \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \in \mathbb{R}^{12\times 12}, \tag{6.44}$$

where $\mathbf{0} \in \mathbb{R}^{3\times 3}$ and $\boldsymbol{I} \in \mathbb{R}^{3\times 3}$.

Correction - Kinematic Measurement

It should be noted that the vision provides us with a lower update rate than the kinematics. Thus, whenever no vision update is available, the kinematic measurement alone is used for the correction of the state. Every time a new kinematic measurement arrives, it is used to correct the first six entries of our state vector. For application on the real rover system, the covariance of the measurement would need to be recovered first. This can be done by experimental evaluation of the difference in commanded and measured positions of the arm.

By rewriting the information on the estimated translation ${}^{C}\hat{\boldsymbol{t}}_F$ and rotation $\boldsymbol{R}(\boldsymbol{q}_{mem})$ between camera and flange by its homogeneous transform ${}^{C}\hat{\boldsymbol{T}}_F$ the innovation introduced by a new incoming kinematic measurement ${}^{C}\boldsymbol{T}_F$ is given by

$$\bar{\boldsymbol{y}}_k = {}^{C}\boldsymbol{T}_F {}^{C}\hat{\boldsymbol{T}}_F^{-1} = \begin{bmatrix} \boldsymbol{\omega} \\ {}^{C}\boldsymbol{t}_{F_k} - \boldsymbol{\omega}\, {}^{C}\hat{\boldsymbol{t}}_{F_{k-1}} \end{bmatrix} \in \mathbb{R}^6, \tag{6.45}$$

where

$$\boldsymbol{\omega} = \log(\boldsymbol{R}(\boldsymbol{q}_F)\boldsymbol{R}(\boldsymbol{q}_{mem})^{-1}) \tag{6.46}$$

is the rotational difference between the current and last time step.

Since the measurement function is given by $\boldsymbol{h}(\boldsymbol{x}_k) = {}^{C}\hat{\boldsymbol{T}}_F$, the measurement matrix for this step is found by calculating its Jacobian, as given in Eq. 6.33. In the following we show its derivation with respect to $\boldsymbol{\omega}_F$ and ${}^{C}\hat{\boldsymbol{t}}_F$, as only these two state variables are considered in the calculation of the innovation for the kinematic measurement update.

We start by defining the decoupled $\mathbf{SO}(3)$ logarithm for the general transformation matrix $\boldsymbol{T}$ [79, p. 48]:

$$\log_d(\boldsymbol{T}) = \begin{bmatrix} \log(\boldsymbol{R}) \\ \boldsymbol{t} \end{bmatrix} \tag{6.47}$$

The derivative of $\boldsymbol{h}(\boldsymbol{x}_k)$ can be defined using the decoupled left increment $\oplus$ [80] and considering the an infinitesimal transformation $\tau = \begin{bmatrix} \boldsymbol{\omega} & \boldsymbol{v} \end{bmatrix}^T$, such that

$$\begin{aligned} \frac{\delta \boldsymbol{h}(\boldsymbol{x})}{\delta {}^{C}\hat{\boldsymbol{T}}_F} &:= \lim_{\tau \to 0} \frac{\log\left(({}^{C}\hat{\boldsymbol{T}}_F \oplus \tau){}^{C}\hat{\boldsymbol{T}}_F^{-1}\right)}{\tau} \\ &= \lim_{\tau \to 0} \frac{\log_d\left(\begin{bmatrix} \exp(\boldsymbol{\omega})\boldsymbol{R} & \boldsymbol{v} + {}^{C}\hat{\boldsymbol{t}}_F \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}^T & -\boldsymbol{R}^T\, {}^{C}\hat{\boldsymbol{t}}_F \\ 0 & 1 \end{bmatrix}\right)}{\tau} \\ &= \lim_{\tau \to 0} \frac{\begin{bmatrix} \boldsymbol{\omega} \\ -\exp(\boldsymbol{\omega}){}^{C}\hat{\boldsymbol{t}}_F + \boldsymbol{v} + {}^{C}\hat{\boldsymbol{t}}_F \end{bmatrix}}{\tau} \end{aligned} \tag{6.48}$$

$$= \lim_{\tau \to 0} \frac{\begin{bmatrix} \boldsymbol{\omega} \\ -\exp(\boldsymbol{I} + [\boldsymbol{\omega}]_\times)^C\hat{\boldsymbol{t}}_F + \boldsymbol{v} + {}^C\hat{\boldsymbol{t}}_F \end{bmatrix}}{\tau}$$
$$= \lim_{\tau \to 0} \frac{\begin{bmatrix} \boldsymbol{\omega} \\ [{}^C\hat{\boldsymbol{t}}_F]_\times + \boldsymbol{v} + {}^C\hat{\boldsymbol{t}}_F \end{bmatrix}}{\tau}$$
$$= \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ [{}^C\hat{\boldsymbol{t}}_F]_\times & \boldsymbol{I} \end{bmatrix}.$$

Note, that we used the first Taylor expansion of the rotational increment $\exp([\boldsymbol{\omega}]_\times) \approx \boldsymbol{I} + [\boldsymbol{\omega}]_\times$.

Thus, the matrix defining the measurement model is given by

$$\boldsymbol{H}_k = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ [{}^C\hat{\boldsymbol{t}}_{F_{k-1}}]_\times & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{6\times 12}. \tag{6.49}$$

Here $\boldsymbol{0} \in \mathbb{R}^{3\times 3}$ and $\boldsymbol{I} \in \mathbb{R}^{3\times 3}$ is the identity matrix. With this, the Kalman gain $\boldsymbol{K}_k$ is calculated as given in Eq. 6.29.
This leads to the state update

$$\boldsymbol{x}_k = \boldsymbol{K}_k \bar{\boldsymbol{y}}_k. \tag{6.50}$$

The error covariance is updated using the standard Kalman Eq. 6.30. Lastly, a reset step [77] is applied to update the auxiliary rotation

$$\boldsymbol{q}_{mem} = \delta\boldsymbol{q}(\hat{\boldsymbol{\omega}}_{F_k})\boldsymbol{q}_{mem} \tag{6.51}$$

and reset the rotation error stored in the state

$$\hat{\boldsymbol{\omega}}_{F_k} = \boldsymbol{0} \in \mathbb{R}^3. \tag{6.52}$$

Since we can not directly observe the velocity component of the state, as both kinematic and vision measurements only supply us with the position of the end-effector, the velocities would stay indefinitely stuck in their initial state when using our motion model given in equations 6.43a- 6.43d. Thus, we compute its "observed" value as done in [81]

$${}^C\boldsymbol{v}_k = \frac{{}^C\boldsymbol{t}_{F_k} - {}^C\hat{\boldsymbol{t}}_{F_{k-1}}}{\Delta t} \tag{6.53}$$

and update the state accordingly.

Correction - Vision Measurement

The correction step applied for an incoming vision measurement follows the same pattern as that of a kinematic measurement. The only differences lie in the calculation of the innovation and the definition of the observation model. The innovation, given a new vision update is

$$\bar{\boldsymbol{y}}_k = {}^{C}\boldsymbol{t}_E - \underbrace{({}^{C}\hat{\boldsymbol{t}}_{F_{k-1}} + (\boldsymbol{R}(\boldsymbol{q}_{mem})\,{}^{F}\hat{\boldsymbol{t}}_{E_{k-1}})}_{\boldsymbol{h}(\boldsymbol{x}_k)} \in \mathbb{R}^3. \tag{6.54}$$

and the matrix of the observation model is again found by derivation of the Jacobian of $\boldsymbol{h}(\boldsymbol{x}_k)$.

By considering an infinitesimal rotation $\boldsymbol{\omega}$ its derivative with respect to $\boldsymbol{\omega}_F$ is defined as

$$\frac{\delta \boldsymbol{h}(\boldsymbol{x}_k)}{\delta \boldsymbol{\omega}_F} := \lim_{\omega \to 0} \frac{{}^{C}\hat{\boldsymbol{t}}_F + \exp([\boldsymbol{\omega}]_\times)\boldsymbol{R}(\boldsymbol{q}_{mem})^F\hat{\boldsymbol{t}}_E - {}^{C}\hat{\boldsymbol{t}}_F - \boldsymbol{R}(\boldsymbol{q}_{mem})^F\hat{\boldsymbol{t}}_E}{\boldsymbol{\omega}}. \tag{6.55}$$

Using the first Taylor expansion of the infinitesimal rotational increment and the antisymmetric property for the multiplication of a vector $\boldsymbol{a}$ with a skewsymmetric matrix $[\boldsymbol{b}]_\times$: $[\boldsymbol{b}]_\times \boldsymbol{a} = -[\boldsymbol{a}]_\times \boldsymbol{b}$, the derivative is obtained by

$$\begin{aligned}
\frac{\delta \boldsymbol{h}(\boldsymbol{x}_k)}{\delta \boldsymbol{\omega}_F} &\approx \lim_{\omega \to 0} \frac{(\boldsymbol{I} + [\boldsymbol{\omega}]_\times)\boldsymbol{R}(\boldsymbol{q}_{mem})^F\hat{\boldsymbol{t}}_E - \boldsymbol{R}(\boldsymbol{q}_{mem})^F\hat{\boldsymbol{t}}_E}{\boldsymbol{\omega}} \\
&= \lim_{\omega \to 0} \frac{\boldsymbol{R}(\boldsymbol{q}_{mem})^F\hat{\boldsymbol{t}}_E + [\boldsymbol{\omega}]_\times \boldsymbol{R}(\boldsymbol{q}_{mem})^F\hat{\boldsymbol{t}}_E - \boldsymbol{R}(\boldsymbol{q}_{mem})^F\hat{\boldsymbol{t}}_E}{\boldsymbol{\omega}} \\
&= \lim_{\omega \to 0} \frac{-[\boldsymbol{R}(\boldsymbol{q}_{mem})^F\hat{\boldsymbol{t}}_E]_\times \boldsymbol{\omega}}{\boldsymbol{\omega}} \\
&= -[\boldsymbol{R}(\boldsymbol{q}_{mem})^F\hat{\boldsymbol{t}}_E]_\times \in \mathbb{R}^{3\times 3}.
\end{aligned} \tag{6.56}$$

For the third component of the state, the derivative is again found by considering an infinitesimal displacement $\boldsymbol{v}$ and given by

$$\begin{aligned}
\frac{\delta \boldsymbol{h}(\boldsymbol{x}_k)}{\delta^F\hat{\boldsymbol{t}}_E} &:= \frac{{}^{C}\hat{\boldsymbol{t}}_F + \boldsymbol{R}(\boldsymbol{q}_{mem})({}^F\hat{\boldsymbol{t}}_E + \boldsymbol{v}) - {}^{C}\hat{\boldsymbol{t}}_F - \boldsymbol{R}(\boldsymbol{q}_{mem})^F\hat{\boldsymbol{t}}_E}{\boldsymbol{v}} \\
&= \frac{\boldsymbol{R}(\boldsymbol{q}_{mem})^F\hat{\boldsymbol{t}}_E + \boldsymbol{R}(\boldsymbol{q}_{mem})\boldsymbol{v}) - \boldsymbol{R}(\boldsymbol{q}_{mem})^F\hat{\boldsymbol{t}}_E}{\boldsymbol{v}} \\
&= \boldsymbol{R}(\boldsymbol{q}_{mem}).
\end{aligned} \tag{6.57}$$

Derivation with respect to ${}^{C}\hat{\boldsymbol{t}}_F$ and ${}^{C}\hat{\boldsymbol{v}}_F$ is trivial and results to

$$\frac{\delta \boldsymbol{h}(\boldsymbol{x}_k)}{\delta^C\hat{\boldsymbol{t}}_F} = \boldsymbol{I} \in \mathbb{R}^{3\times 3} \tag{6.58}$$

and

$$\frac{\delta \boldsymbol{h}(\boldsymbol{x}_k)}{\delta^C\hat{\boldsymbol{v}}_F} = \boldsymbol{0} \in \mathbb{R}^{3\times 3}, \tag{6.59}$$

respectively.

The observation model is thus given by

$$\boldsymbol{H}_k = \begin{bmatrix} -[\boldsymbol{R}(\boldsymbol{q}_{mem})^F\hat{\boldsymbol{t}}_E]_\times & \boldsymbol{I} & \boldsymbol{R}(\boldsymbol{q}_{mem}) & \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{3\times 12}. \tag{6.60}$$

From here on, the same steps as for the kinematic measurement correction given above are carried out to update the state and reset the rotations, except for the velocity update.

6.3.4. Validation Gate

So far, the ES-EKF provides us with a smoothed pose estimate. However, the sensor readings obtained by the vision pipeline can contain faulty measurements, motivating us to include an additional outlier detection within the ES-EKF. This might happen, for example, if there is an additional pattern of bright points in the image, close to the real marker array, e.g. because of specular reflections of the LEDs on the last link of the robot arm as it may cause the detection algorithm to wrongly detect those instead of the marker array.

To increase the robustness of the filter, we perform a hypothesis test within the vision measurement correction step, to ensure that the observation received by the filter is compatible with the underlying model, also referred to as the null hypothesis [82]. For the hypothesis test we use a measurement validation gate [83] based on the normalized estimation error squared (NEES) d^2. For this purpose, we use the squared Mahalanobis distance of the measurement innovation [83]

$$d^2 = \bar{\boldsymbol{y}}_k^T (\boldsymbol{H}_k \boldsymbol{P}_k \boldsymbol{H}_k^T + \boldsymbol{\Sigma}_{tr})^{-1} \bar{\boldsymbol{y}}_k, \tag{6.61}$$

to check whether the observation is compatible with our null hypothesis, or if it is faulty and should be rejected. The NEES hereby follows a Chi-squared distribution with m DOF since $\bar{\boldsymbol{y}}_k \in \mathbb{R}^m$.

The validation gate defines a region of acceptance below a bounding value γ of the Chi-squared distribution for a given confidence level $\alpha \in [0, 1]$, such that $100(1-\alpha)\%$ of true measurements are rejected [83]. The validity criterion for an observation is thus [83]

$$V(m, z) = \{z : d^2 \leq \gamma\}. \tag{6.62}$$

For the translation measurement with $m = 3$ DOF, we chose a confidence level of $\alpha = 0.95$ to ensure reliant outlier detection. The bounding value is $\gamma(\alpha, m) = 7.81$, as given in [84].

After application of the Kalman filter, the position corrected result of the vision pipeline is given by the transformation ${}^F\boldsymbol{T}_E = \begin{bmatrix} \boldsymbol{q}_I & {}^F\boldsymbol{t}_{E_k} \end{bmatrix}$, where $\boldsymbol{q}_I$ is the identity quaternion. This transform is subsequently used to update the transformation tree.

6.3.5. Evaluation - ESEKF

Since the real rover platform was not available for testing, the ES-EKF was evaluated and tuned in simulation (Sec. 7.4). The reason for tuning the ES-EKF in the simulation environment and not directly on the prerecorded data-sets was that it is initially planned to be used together with the the PBVS scheme, described in depth chapter 7, for first experiments in simulation. Once the real rover system is available and correct, non-lagging ground-truth data is available, tuning and evaluation of the ES-EKF will need to be carried out on the rover system.

To account for the kinematic imprecision of the real manipulator, a constant error offset of 20 mm was added to both the X, and Y-axis of the kinematic measurements of the end-effector frame. The operating frequency of the filter was set to $f = 100\,\text{Hz}$. The noise covariance matrix of the kinematic measurements was tuned manually to $\boldsymbol{S} = \text{diag}(1\text{e}^{-9})$. This considerably small noise level has to be viewed as a strong idealization, due to the fact that the simulated kinematic measurements are almost noise free.

The process noise covariance matrix was set to $\boldsymbol{Q} = \text{diag}(0.001)\Delta t$, where $\Delta t = \frac{1}{f}$. This value gave the best smoothest results, without becoming too slow to respond to new measurement inputs.

Fig. 6.7 shows the performance evaluation of the ES-EKF in simulation, by comparing the normed error of the vision corrected position estimate before, and after application of the filter, relative to the simulated ground truth.

At the beginning of the recording, the end-effector is tilted by approximately 45° to the optical axis of the camera. It is then moved to a pose perpendicular to the optical axis in two consecutive steps. With this in mind, it is apparent that the tilt of the marker array considerably influences the noisiness of the visual position estimate. Additionally, it has to be pointed out that in this particular experiment an error in the definition of the radius for the geometric model of the active marker array, used to retrieve the 3D pose of the end-effector, led to an unexpectedly high position error along the optical axis of the camera. Which in turn led to a larger normed error than in the initial evaluation in section 6.2.2. The model missmatch is due to the fact that the simulated LED array does not have the exact same diameter as that of the real rover. Unfortunately, this was only discovered and corrected at a later point. The focus of this evaluation therefore lies on the smoothing properties of the filter.

The first 5 s as well as the time between 10 s and 20 s of operation illustrate the smoothing abilities of the ES-EKF during steady state of the end-effector. It can be seen that the filter successfully reduces the noise introduced by the vision measurement, showing a roughly constant position error.

The sampling periods between 5 s to 10 s and 20 s to 25 s, show the behavior for slow and fast changes in the vision error, respectively. In both cases, the end-effector was moved vertically, while ensuring its visibility in the image stream. We can see that the

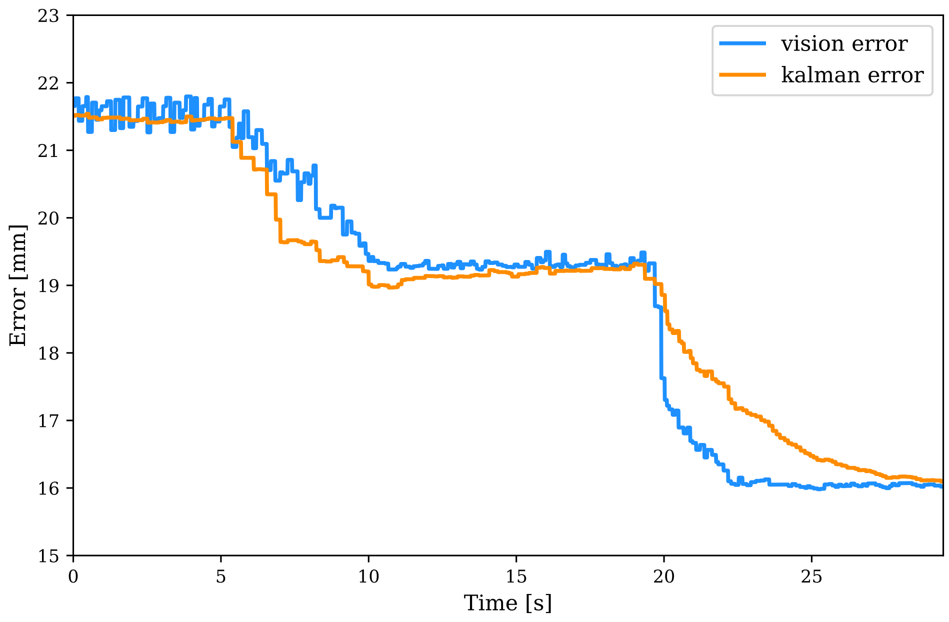


Figure 6.7.: Normed errors of the vision-based position correction pipeline before (blue) and after application of the ES-EKF (orange), as tested in simulation. Not depicted is the constant normed error of 28 mm for the kinematic measurements.

filter still reduces the noise in the estimate. However, it shows a slow response to fast changes in the vision error. This is due to our choice for Q, which leads to a trade-off between the smoothing ability and dynamics of the filter. For use with the real rover platform, a different tuning of Q might be necessary to improve the dynamic behavior of the filter, depending on the desired velocity of the end-effector movement.

In conclusion, we showed that the application of the ES-EKF reduces the noise of the filtered signal and provides us with a more stable input signal for our visual servo scheme, than the vision-based correction alone. However, its smoothing abilities come at the price of a reduced responsiveness to dynamic changes in the measurements.

7. Improving Payload Docking Through Visual Servoing

The previous chapter showed that the use of visual data from the LRU2s on-board cameras yields an improved estimate of the docking interface position compared to using the forward kinematics measurements alone. We can continue and leverage this information to improve the precision of the rover's payload docking process by using visual servoing to supply the on-board controllers of the manipulator with an adequate input signal, generated on the basis of the vision corrected end-effector pose estimate.

As the vision pipeline provides us with a 3D pose estimate of the end-effector pose, we will employ two PBVS based servoing schemes for the approach and docking phase, respectively. Using a switching controller, a classical PBVS scheme is used during the approach phase in free space, while switching to a PBVS based hybrid impedance visual servoing scheme once the end-effector is in immediate proximity of its coupling partner. The classical PBVS is further extended to improve the stability of the system in the vicinity of singularities and to comply to the position and velocity limitations inherent to the manipulator to ensure its safe operation.

We begin this chapter with the formulation of servoing requirements to consider for our system, followed by the detailed description of the methods applied during approach and docking as well as the utilized arm controllers.

7.1. Requirements

To be able to precisely dock to a payload box, the visual servoing schemes must meet several requirements. These can be split up into general requirements, applying to both the approach- and the docking phase, and special requirements to consider in the latter one, where we establish direct contact to the payload box. In general, both visual servo approaches should:

- be able to make use of the 3D pose estimates obtained from the tracking pipeline,
- yield high positional accuracy,
- be safe and adhere to system limitations, e.g. maximum joint speeds and joint position limits,

- be stable within the manipulator's work space, and
- provide an adequate input signal to the on-board controllers of the manipulator.

During the docking phase, when the end-effector comes into contact with the payload target, the visual servo should additionally:

- ensure a save environment interaction behavior between the manipulator and its environment, and
- keep the end-effector correctly positioned within the docking plane.

7.2. Approach Phase

The goal of the approach phase is to bring the docking interface into a pose that allows it to easily dock to its coupling partner mounted on the payload box. For this purpose, we position the docking interface at a predefined, small distance Δz in front of its coupling partner, and align it such that only a forward translation along the longitudinal axis Z_{TCP} of its tool center point (TCP) frame is needed to establish contact to the coupling partner during the docking phase, see Fig. 7.1. The TCP frame is a virtual reference frame, whose X-Y plane coincides with that of the coupling partner, once the docking procedure is completed. It is defined relative to the end-effector frame by a constant translation of 12 cm and rotation of $+90°$ along the Z-axis of the vision corrected end-effector frame. Since the docking interface has a cylindrical form, we can ignore the

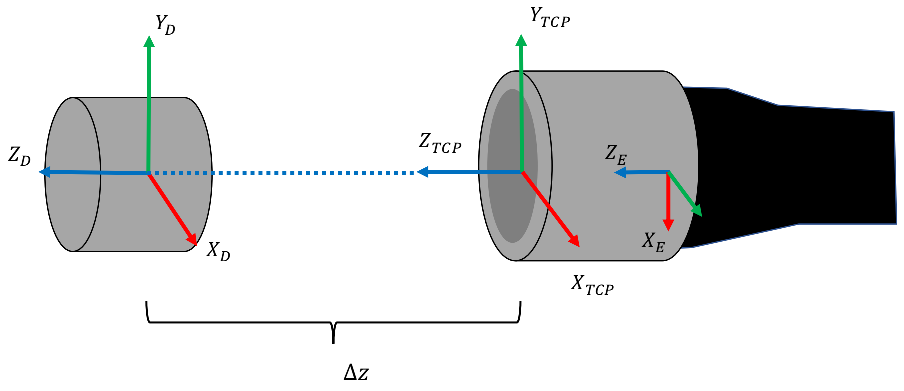


Figure 7.1.: Alignment of the docking target and tool center point frame at the end of the approach phase. O_E marks the center of origin of the vision corrected end-effector frame.

rotation around Z_{TCP} during the alignment task, leaving us with five DOF to control. As we will see in Sec. 7.3, of these five DOF especially the correct positioning in the coupling target's X-Y plane is of importance during the visual servoing, as the position tolerance during the docking is restricted to ± 1 cm.

We have learned in Sec. 2.1 that the manipulator offers four different control modes: two motion controllers and two impedance controllers, each pair consisting of one task space and one joint space controller. For the above reasons, we will be applying a motion controller during the approach phase. While the Cartesian position controller would present the most straight forward way to position the end-effector in a predefined Cartesian location, its current implementation is not operational. Thus, we will instead make use of the joint space controller.

7.2.1. Joint Space Position Controller

The available joint space controller is a position controller, following a proportional-integrational (PI) velocity control scheme. To describe it, we first need to consider the dynamic model of the manipulator in the joint space [85]

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}, \tag{7.1}$$

where for our six DOF manipulator $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}} \in \mathbb{R}^6$ are the vectors of joint positions, velocities and accelerations, $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{6\times 6}$ is the positive definite symmetric manipulator inertia matrix, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \in \mathbb{R}^6$ is the vector of Coriolis and centrifugal forces, $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^6$ is the vector of gravitational forces and $\boldsymbol{\tau} \in \mathbb{R}^6$ is the vector of resulting joint torques [85]. Note, that this is a simplified model, which does not consider friction or other time varying disturbances. The torque can be rewritten in terms of the servo-amplifier input voltages $\boldsymbol{u} \in \mathbb{R}^6$ for all joints [85]

$$\begin{aligned} \boldsymbol{\tau} &= \boldsymbol{K}_{mot}\boldsymbol{i}_{mot} \\ &= \boldsymbol{K}_{mot}\boldsymbol{K}_{servo}\boldsymbol{u}, \end{aligned} \tag{7.2}$$

where $i_{mot} \in \mathbb{R}^6$ is the vector of motor currents, $\boldsymbol{K}_{mot} \in \mathbb{R}^{6\times 6}$ is the diagonal matrix containing the motor constants relating the input current to the torque output and $\boldsymbol{K}_{servo} \in \mathbb{R}^{6\times 6}$ is another diagonal matrix containing the servo-amplifier gains [85]. With help of Eq. 7.2, we can rewrite the dynamic model 7.1 as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{K}_{mot}\boldsymbol{K}_{servo}\boldsymbol{u}. \tag{7.3}$$

The classical PI joint velocity controller is given by [85] as

$$\boldsymbol{u} = \boldsymbol{K}_P\dot{\boldsymbol{e}} + \boldsymbol{K}_I\boldsymbol{e}, \tag{7.4}$$

with $\boldsymbol{K}_P \in \mathbb{R}^6$ and $\boldsymbol{K}_I \in \mathbb{R}^6$ being diagonal, positive definite matrices, $\dot{\boldsymbol{e}} = \dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}$ being the error between the desired $\dot{\boldsymbol{q}}_d$ and current $\dot{\boldsymbol{q}}$ joint velocities and the integral of the error measure $\boldsymbol{e} = \int_0^t(\dot{\boldsymbol{e}}(\rho))d\rho$.

Substituting the PI control law 7.4 into the equation of the robot dynamics 7.3 and setting $\dot{\boldsymbol{q}}(t) = (\dot{\boldsymbol{q}}_d(t) - \dot{\boldsymbol{e}}(t))$, we can write [85]

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}(\boldsymbol{q}(t))^{-1}[\boldsymbol{M}(\boldsymbol{q}(t)\ddot{\boldsymbol{q}}_d + \boldsymbol{C}(\boldsymbol{q}(t),\dot{\boldsymbol{q}}(t))\dot{\boldsymbol{q}}(t) + \boldsymbol{g}(\boldsymbol{q}) - \boldsymbol{K}\boldsymbol{K}_P\dot{\boldsymbol{e}} - \boldsymbol{K}\boldsymbol{K}_I\boldsymbol{e}]. \tag{7.5}$$

By defining the control objective to be the reaching of a constant $\dot{\boldsymbol{q}}(t)$ with $\ddot{\boldsymbol{q}}(t) = 0$ Eq. 7.5 can be rewritten as

$$\mathbf{0} = \boldsymbol{M}(\boldsymbol{q}(t))^{-1}[\boldsymbol{M}(\boldsymbol{q}(t)\ddot{\boldsymbol{q}}_d + \boldsymbol{C}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t))\dot{\boldsymbol{q}}(t) + \boldsymbol{g}(\boldsymbol{q}) - \boldsymbol{K}\boldsymbol{K}_P\dot{\boldsymbol{e}} - \boldsymbol{K}\boldsymbol{K}_I\boldsymbol{e}]. \tag{7.6}$$

The controller implementation used for our robot arm defines the controller input $\dot{\boldsymbol{q}}_d(t)$ in terms of two required user inputs: The commanded joint velocities $\dot{\boldsymbol{q}}_{cmd}(t)$ and positions $\boldsymbol{q}_{cmd}(t)$. In addition, it considers the current joint positions to yield the desired velocity

$$\dot{\boldsymbol{q}}(t)_d = \dot{\boldsymbol{q}}_{cmd}(t) + (\boldsymbol{q}_{cmd}(t) - \boldsymbol{q}(t))\boldsymbol{K}_{pos}, \tag{7.7}$$

where $\boldsymbol{K}_{pos}$ is a constant. It can be used to neglect position related term of the equation, by setting $\boldsymbol{K}_{pos} = 0$. This means, all we need to supply to control the position of the manipulator joints with our visual servo, is a vector of commanded joint velocities that will drive the end-effector towards its desired docking position.

7.2.2. Position-Based Visual Servo

The general objective of visual servoing is to minimize an error function $\boldsymbol{e}_x(t)$ between a set of tracked visual features and their desired pose either in three-dimensional space or directly in the image. A definition of the error function is given by [86] as

$$\boldsymbol{e}_x(t) = \boldsymbol{s}(\mathbf{m}(t), \mathbf{a}) - \boldsymbol{s}^*, \tag{7.8}$$

where $\mathbf{m}(t)$ is a vector containing image measurements, such as the image space coordinates of the visual features. This is used to compute $\boldsymbol{s}(\mathbf{m}(t), \mathbf{a})$, a vector of k visual features, where $\mathbf{a}$ is a set of parameters representing additional system knowledge, such as a 3D model of the tracked object or the camera's intrinsic parameters [51]. Lastly, $\boldsymbol{s}^*$ is the vector of the desired feature coordinates which we wish to approach. The design of $\boldsymbol{s}$ varies, depending on the chosen servoing scheme (IBVS or PBVS).

For PBVS, the feature $\boldsymbol{s}$ is defined by a relative transformation between the desired and current pose of the manipulator's end-point in the task space. In our case, this is given by the transform ${}^{D}\boldsymbol{T}_{TCP} = ({}^{C}\boldsymbol{T}_D)^{-1}\,{}^{C}\boldsymbol{T}_{TCP}$ between the pose estimate of the payload box ${}^{C}\boldsymbol{T}_D$ as received from the AprilTag detection, and that of the TCP frame ${}^{C}\boldsymbol{T}_{TCP}$, implicitly given by our vision corrected end-effector pose estimate, such that

$$\boldsymbol{s} = \begin{bmatrix} {}^{D}\boldsymbol{t}_{TCP} \\ \theta\boldsymbol{u} \end{bmatrix} \in \mathbb{R}^6, \tag{7.9}$$

with ${}^{D}\boldsymbol{t}_{TCP}$ being the translational and $\theta\boldsymbol{u}$ the rotational difference between the two poses, expressed in axis/angle coordinates. Since we ignore the rotation along the docking interface's Z-axis, we can directly set

$$\theta\boldsymbol{u} = \begin{bmatrix} \omega_x \\ \omega_y \\ 0 \end{bmatrix}. \tag{7.10}$$

To align the frames, both ${}^{D}\boldsymbol{t}_{TCP}$ and $\theta\boldsymbol{u}$ need to be $\mathbf{0}$ and therefore $\boldsymbol{s}^* = \mathbf{0}$ and $\boldsymbol{e}_x = \boldsymbol{s}$.

To drive the pose error towards zero using the available velocity controller, we first need to define the relationship between the TCP velocity ${}^{TCP}\boldsymbol{v} = \begin{bmatrix}\mathbf{v} & \boldsymbol{\omega}\end{bmatrix}^T$, with $\mathbf{v}$ and $\boldsymbol{\omega}$ being its translational and rotational components, given with respect to the coordinates of the TCP frame, and the time variation of the error

$$\dot{\boldsymbol{e}}_x = \boldsymbol{L}\,{}^{TCP}\boldsymbol{v}, \tag{7.11}$$

where $\boldsymbol{L} \in \mathbb{R}^{k\times 6}$ is known as the *interaction matrix* [18]. The specific composition of the interaction matrix depends on the camera configuration of the system [18]. For our eye-to-hand case, we define it as

$$\boldsymbol{L} = \begin{bmatrix} -\boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & -\boldsymbol{L}_{\theta u} \end{bmatrix}, \tag{7.12}$$

where $\boldsymbol{I} \in \mathbf{R}^{3\times 3}$ and $\boldsymbol{L}_{\theta u}$ is given by [86] as

$$\boldsymbol{L}_{\theta u} = \boldsymbol{I} - \frac{\theta}{2}[\boldsymbol{u}]_\times + \left(1 - \frac{\operatorname{sinc}\theta}{\operatorname{sinc}^2\frac{\theta}{2}}\right)[\boldsymbol{u}]_\times^2 \in \mathbb{R}^{3\times 3}, \tag{7.13}$$

with $\operatorname{sinc}\theta$, being the sine cardinal defined such that $\theta\operatorname{sinc}\theta = \sin\theta$ and $\operatorname{sinc}0 = 1$.

Rewriting Eq. 7.11 by substituting $\dot{\boldsymbol{e}}_x = -\lambda\boldsymbol{e}_x$ [18] with $\lambda \geq 0$, thus ensuring an exponential decrease of the error, the calculation of ${}^{TCP}\boldsymbol{v}$ results to [86]

$${}^{TCP}\boldsymbol{v} = -\lambda\boldsymbol{L}^{-1}\boldsymbol{e}_x. \tag{7.14}$$

In order to obtain the vector of desired joint velocities for the controller, two last steps are necessary to transform the Cartesian TCP velocity to the joint space. To calculate the joint space velocities by using the manipulator's body Jacobian, as introduced in Sec. 4.2.4, we first need to transform the velocity vector to the coordinates of the manipulator's base frame B. To do so, we apply the spacial transformation Jacobian [72, p.232]

$${}^{B}\boldsymbol{J}_{TCP} = \begin{bmatrix} {}^{B}\boldsymbol{R}_{TCP} & \mathbf{0} \\ \mathbf{0} & {}^{B}\boldsymbol{R}_{TCP} \end{bmatrix}, \tag{7.15}$$

mapping the vector to the base frame

$${}^{B}\boldsymbol{v} = {}^{B}\boldsymbol{J}_{TCP}\,{}^{TCP}\boldsymbol{v}. \tag{7.16}$$

Lastly, using the body Jacobian ${}^{B}\boldsymbol{J}$ the input to the joint space velocity control law can be calculated by

$$\dot{\boldsymbol{q}} = {}^{B}\boldsymbol{J}_e^{+}\,{}^{B}\boldsymbol{v}. \tag{7.17}$$

The calculations applied in Eq. 7.12 to 7.17 can be condensed to a single step, by defining a task Jacobian $\boldsymbol{J}_e$ of the form [87]

$$\boldsymbol{J}_e = \boldsymbol{L}\,({}^{B}\boldsymbol{J}_{TCP})^{T}\,{}^{B}\boldsymbol{J} \tag{7.18}$$

and thus directly relating the error to the desired joint velocities by

$$\dot{\boldsymbol{q}}_d = -\lambda \boldsymbol{J}_e^{+}\boldsymbol{e}. \tag{7.19}$$

It has to be pointed out, that deducing $\dot{\boldsymbol{q}}_d$ from the desired TCP velocity and using Eq. 7.17 is generally not equivalent to using the task Jacobian 7.19 [87], as when $\boldsymbol{J}_e^{+} \neq {}^{B}\boldsymbol{J}^{+}\,{}^{B}\boldsymbol{J}_{TCP}\,\boldsymbol{L}^{-1}$ both control schemes will induce slightly different manipulator trajectories [87]. It is therefore important to note, that we will be using the definition of the servoing law given in Eq. 7.19 in the remainder of this chapter, as it allows us to formulate subtasks more easily, as we will see in Sec. 7.2.2. In the following, we will discuss several additions made to the standard visual servoing law given in Eq. 7.19, meant to improve its behavior during the approach phase.

Improving Numerical Stability: Damped Least Squares Jacobian

Our manipulator presents us with three different types of singular configurations [88]: The first type is the boundary singularity, which is encountered when the arm is fully extended. As the limit of its work space is reached, the manipulator can not move any further in the direction that it is stretched out in. The second form of singularity is the wrist-over-base [88] case, where the wrist is aligned with the axis of the first joint. The last type is the wrist alignment singularity [88], which occurs when the axes of joints four and six are aligned. This causes the end-effector to lose the rotational DOF around that axis. Fig. 7.2 shows all three cases.

As we will see in Sec. 7.5.2, first evaluations of our methods showed that using the Moore-Penrose pseudo inverse $\boldsymbol{J}_e^{+}$ of the task Jacobian in Eq. 7.19, leads to numerical stability problems in the neighborhood of singularities. While the pseudo inverse method yields a stable behavior *in* a singular configuration [58], close-to-singular configurations lead to large fluctuations in the commanded joint velocities.

To avoid this issue, we instead implement the damped least-squares (DLS) method for the inversion of $\boldsymbol{J}_e$, to determine the desired joint velocities in a numerically stable fashion [58]. Applied to the computation of $\dot{\boldsymbol{q}}_d$, the DLS approach corresponds to the calculation of the Levenberg-Marquart update step introduced in Sec. 4.3.4, such that

$$\dot{\boldsymbol{q}}_d = -\lambda \boldsymbol{J}_e^{T}(\boldsymbol{J}_e^{T}\boldsymbol{J}_e + d^2\boldsymbol{I})^{-1}\boldsymbol{e}, \tag{7.20}$$

where d is the damping constant that needs to be tuned according to the given application. The DLS Jacobian is then given by

$$\boldsymbol{J}_e^{\#} = \boldsymbol{J}_e^{T}(\boldsymbol{J}_e^{T}\boldsymbol{J}_e + d^2\boldsymbol{I})^{-1} \tag{7.21}$$

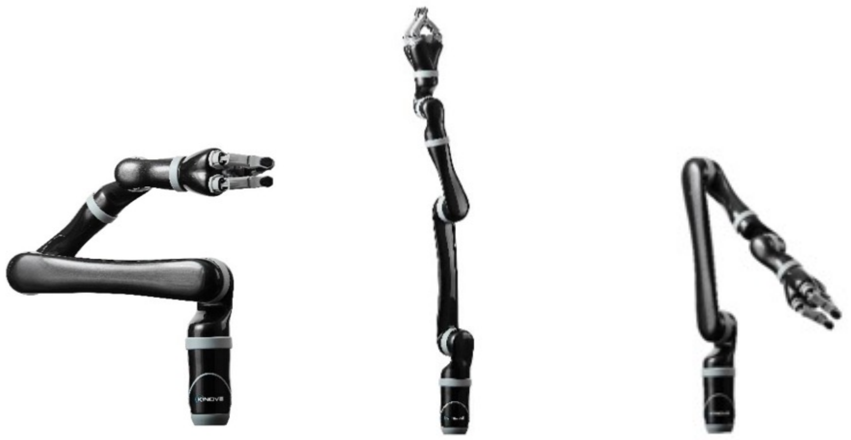


Figure 7.2.: Singular configurations of the Jaco2 arm, as given in [88]. From left to right: Wrist-over-base, boundary and wrist alignment singularity.

and we can rewrite Eq. 7.19 as

$$\dot{\boldsymbol{q}}_d = -\lambda \boldsymbol{J}_e^{\#} \boldsymbol{e}. \tag{7.22}$$

The influence of the damping constant is best understood by considering the singular value decompositions of $\boldsymbol{J}_e^+$ and $\boldsymbol{J}_e^{\#}$ as given in [58] by:

$$\boldsymbol{J}_e^+ = \sum_{i=1}^{r} \sigma_i^{-1} \mathbf{v}_i \boldsymbol{u}_i^T \tag{7.23}$$

and

$$\boldsymbol{J}_e^{\#} = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + d^2} \mathbf{v}_i \boldsymbol{u}_i^T, \tag{7.24}$$

where r indicates the number of rows of the Jacobian, σ_i are its singular values and $\mathbf{v}_i$ and $\boldsymbol{u}_i$ are the left-singular and right-singular vectors of σ_i, respectively. A singularity is characterized by $\sigma_i = 0$.

By comparing Eq. 7.23 and Eq. 7.24, we can see that in the case of configurations far away from singularities, where $\sigma_i >> d$ the behavior of the DLS method is comparable to that of the pseudo inverse method [58]. However, when approaching a singular configuration, where σ_i becomes of similar magnitude as d, the DLS method ensures the numerical stability of $\boldsymbol{J}_e^{\#}$, as for any $d > 0$, $\frac{\sigma_i}{\sigma_i^2 + d^2} \rightarrow 0$ as $\sigma_i \rightarrow 0$ [58].

The damping constant needs to be chosen such, that it is large enough to ensure that the manipulator behaves smoothly near singular configurations. At the same time, it should not be chosen too large, as that would slow down the convergence towards the desired pose [58]. We found a value of $d = 0.03$ to be well suited for our use-case.

Increasing Convergence Speed: Adaptive Gain

The standard servoing law 7.19 uses a constant gain λ. While this ensures the exponential decoupled decrease of the error [86], it also means that, while the commanded joint velocity will be high when the end-effector is far from its goal pose, it will decrease the closer the end-effector gets. This leads to slow convergence of the system.

To reduce the convergence time, we instead use an adaptive gain as proposed in [43]:

$$\lambda(x) = (\lambda_0 - \lambda_\infty) \exp\left(-\frac{\lambda'_0}{\lambda_0 - \lambda_\infty x}\right) + \lambda_\infty, \tag{7.25}$$

where

- $\lambda_0 = \lambda(0)$ is the gain for $||\boldsymbol{e}|| \to 0$,
- $\lambda_\infty = \lambda_{||\boldsymbol{e}|| \to \infty} \lambda(||\boldsymbol{e}||)$ is the gain for very large $||\boldsymbol{e}||$,
- λ'_0 is the slope of λ at $||\boldsymbol{e}|| = 0$ and
- $x = ||\boldsymbol{e}||_\infty$ is the infinity norm of the current error.

The values for $\lambda_0, \lambda_\infty$ and λ'_0 have to be set manually and need to be tuned according to the specific application. The correct tuning is important to ensure the stability and convergence of the system: If the value for λ_0 is set too high, the end-effector will be oscillating around its goal position, while if it is too low, the convergence will be very slow. If λ_∞ is chosen too high, the arm might move too fast for the features to be tracked.

The tuning procedure for $\lambda_0, \lambda_\infty$ and λ'_0 will be discussed more in detail in Sec. 7.5.1.

Ensuring Smooth Task Transitions: Continuous Sequencing

When switching to the visual servoing task, we are presented with the problem of possible discontinuities at the beginning of the task [89]. Imagine the end-effector of our arm being positioned at a large distance from the docking target at the moment of switching: Because of the direct relationship between $\dot{\boldsymbol{q}}$ and $\boldsymbol{e}$ in Eq. 7.19, a large displacement error will result in an abrupt jump in the commanded joint velocity, e.g., from zero to maximum tolerable servoing speed at time $t = 0$, surpassing the joints acceleration limits. Thus, a supplementary term has been added to the standard control law 7.19, as suggested by [89], to ensure a smooth transition at the beginning of the visual servoing task and to avoid velocity discontinuities due to abrupt changes in the command:

$$\dot{\boldsymbol{q}} = -\lambda(x)\boldsymbol{J}_e^{\#}\boldsymbol{e} + \underbrace{\lambda(x)\boldsymbol{J}_{e(0)}^{\#}\boldsymbol{e}(0)\exp(-\mu t)}_{\textit{continuous sequencing}}, \tag{7.26}$$

where [89]:

- $\boldsymbol{J}^{\#}_{e(0)}\boldsymbol{e}(0)$ is the value of $\boldsymbol{J}^{\#}_{e}\boldsymbol{e}$ at time $t = 0$,
- μ is a new gain constant and
- t is the time since start in seconds.

Adding Subtasks: The Large Projection Operator

So far our visual servo only follows one main task: driving $\boldsymbol{e}$ towards zero. However, it is often necessary to consider additional constraints concerning, e.g., joint position limits or obstacle avoidance, to prevent damaging the manipulator during the servoing. The classical gradient projection method [90] offers a way to project such secondary constraints on the null-space of the primary task without affecting the nullification of the error. This is done by defining a new, combined task function [90]

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_1 - \boldsymbol{P}_e\boldsymbol{g}, \tag{7.27}$$

where $\dot{\boldsymbol{q}}_1$ represents the main task as given in Eq. 7.26, $\boldsymbol{g}$ contains the motion defined by the secondary task to be applied, and

$$\boldsymbol{P}_e = (\boldsymbol{I} - \boldsymbol{J}^{\#}_e\boldsymbol{J}_e) \tag{7.28}$$

is the projection operator on the null-space of $\boldsymbol{J}_e$, satisfying the condition that injection of $\boldsymbol{g}$ does not interfere with the primary task [90].

Using this projection operator for our manipulator with $n = 6$ DOF to perform a task with $m = 5$ DOF, leaves us with a nullspace of only $m - n = 1$ DOF for the introduction of a secondary task. Running into a singular configuration would decrease this number even further. However, we might wish to be able to use more than that, e.g., in the case of two or more joints approaching their limits or when being in a (near) singular configuration. Thus, we decided to apply a different implementation of $\boldsymbol{P}_e$, given by [91], which instead defines the *large projection operator* on the basis of the norm of the total error $||e||$. This operator has the advantage that it is always at least of rank $n - 1$ [91], and thus applicable even if the main task is of full rank or two or more constraints need to be applied simultaneously. It is given by [91] as

$$\boldsymbol{P}_{||e||} = \boldsymbol{I} - \frac{1}{\boldsymbol{e}^T\boldsymbol{J}_e\boldsymbol{J}_e^T\boldsymbol{e}}\boldsymbol{J}_e^T\boldsymbol{e}\boldsymbol{e}^T\boldsymbol{J}_e. \tag{7.29}$$

Joint Limits Avoidance with the Large Projection Operator

The Jaco arm uses both revolute and continuous joints. The difference between the two types lies in the hard upper and lower position limits for the actuators in the revolute case. These position limits, given in Tab. 7.1, need to be taken into account when supplying the velocity input to the low level joint controller, to prevent damaging the actuators. By using the large projection operator given in Eq. 7.29, the objective of joint

i	1	2	3	4	5	6
q_i^{max} [°]	720	310	341	720	720	720
q_i^{min} [°]	-720	50	19	-720	-720	-720

Table 7.1.: Upper q_i^{max} and lower q_i^{min} position limits of joints $i = 1 \dots 6$ in degree. Limits for joints J_2 and J_3 correspond to those given in [88, p.44]. Positions of the continuous joints J_1, J_4, J_5, J_6 have been limited to prevent any excessive rotation.

avoidance can be formulated as an additional task to be injected into the main visual servoing task. For this purpose, we apply the concept presented in [91], which is given as follows:

We start by defining an additional set of *soft position limits* $q_{l_0 i}^{min}$ and $q_{l_0 i}^{max}$, see Fig. 7.3, indicating a lower and upper bound within which the configuration of the manipulator is deemed safe. They are given by

$$q_{l_0 i}^{min} = q_i^{min} + \rho(q_i^{max} - q_i^{min}), \tag{7.30}$$

$$q_{l_0 i}^{max} = q_i^{max} + \rho(q_i^{max} - q_i^{min}), \tag{7.31}$$

where $\rho \in [0, \frac{1}{2}]$ is a safety constant [91], and in our case $\rho = 0.1$.

In the next step, we define the new servoing law

$$\dot{\boldsymbol{q}}_d = \dot{\boldsymbol{q}}_1 + \dot{\boldsymbol{q}}_2, \tag{7.32}$$

where

$$\dot{\boldsymbol{q}}_2 = \sum_{i=1}^{n} \dot{\boldsymbol{q}}_2^i \tag{7.33}$$

is the joint velocity vector of the joint limits avoidance task and

$$\dot{\boldsymbol{q}}_2^i = -\lambda_{sec_i} \lambda_{l_i} \boldsymbol{P}_{||e||} \boldsymbol{g}_i^l \tag{7.34}$$

is the avoidance task for a single joint i. The vector $\boldsymbol{g}_i^l$ is an *activation and sign function*, indicating whether the current joint configuration is within the soft joint limits, or not. It is defined by considering each joint $j \in 1, ..., 6$, where

$$\boldsymbol{g}_i^l[j] = \begin{cases} -1, & \text{if } q_i < q_{l_0 i}^{min} \text{ and } i = j \\ 1, & \text{if } q_{l_0 i}^{max} < q_i \text{ and } i = j \\ 0, & \text{otherwise.} \end{cases} \tag{7.35}$$

Before defining the gain functions λ_{sec_i} and λ_{l_i} we introduce an additional set of upper and lower *safety position limits* $q_{l_1 i}^{max}$ and $q_{l_1 i}^{min}$, indicating the threshold at which the joint limits avoidance task is fully injected onto the main task. They are given by

$$q_{l_1 i}^{min} = q_i^{min} + \gamma\rho(q_i^{max} - q_i^{min}), \tag{7.36}$$

$$q_{l_1 i}^{max} = q_i^{max} + \gamma\rho(q_i^{max} - q_i^{min}), \tag{7.37}$$

where $\gamma = 0.8$.

Considering these safety limits, the *tuning gain function* λ_{sec_i} is given by

$$\lambda_{l_i(q_i)} = \begin{cases} 1, & \text{if } q_i < q_{l_1 i}^{min} \text{ or } q_{l_1 i}^{max} < q_i \\ \lambda_{li}^{min}(q_i), & \text{if } q_{l_1 i}^{min} \leq q_i \leq q_{l_0 i}^{min} \\ \lambda_{li}^{max}(q_i), & \text{if } q_{l_0 i}^{max} \leq q_i \leq q_{l_1 i}^{max} \\ 0, & \text{if } q_{l_0 i}^{min} < q_i < q_{l_0 i}^{max} \end{cases}, \tag{7.38}$$

where $\lambda_{li}^{min}(q_i)$ and $\lambda_{li}^{max}(q_i)$ are two sigmoid functions, acting as scaling parameters to control the injection of the joint avoidance task and defined by

$$\lambda_{li}^{max}(q_i) = \frac{1}{1 + \exp\left(-12\frac{q_i - q_{l_0 i}^{max}}{q_{l_1 i}^{min} - q_{l_0 i}^{max}} + 6\right)} \tag{7.39}$$

and

$$\lambda_{li}^{min}(q_i) = \frac{1}{1 + \exp\left(-12\frac{q_i - q_{l_0 i}^{min}}{q_{l_1 i}^{min} - q_{l_0 i}^{min}} + 6\right)}. \tag{7.40}$$

Lastly, λ_{sec_i} is an *adaptive gain function*

$$\lambda_{sec_i} = \begin{cases} (1+\lambda_i)\frac{|\dot{q}_1[i]|}{(\boldsymbol{P}_{||e||}\boldsymbol{g}_i^l)[i]}, & \text{if } \boldsymbol{g}_i^l[i] \neq 0 \\ 0, & \text{otherwise} \end{cases}, \tag{7.41}$$

which is used in the vicinity of a joint limit, to regulate the magnitude of $|\dot{\boldsymbol{q}}_2^i[i]|$ such that the corresponding component of the main task is compensated by ensuring that $|\dot{\boldsymbol{q}}_2^i[i]| > |\dot{\boldsymbol{q}}_1^i[i]|$ when approaching the hard upper and lower joint limits.

For a more in depth description, and the extension to the inclusion of tertiary tasks, the interested reader is referred to [91].

Enforcing Joint Velocity Limits

Lastly, to prevent any excessive velocity commands from being send to the arm controller by our PBVS, the last step of our servoing pipeline is the enforcement of the velocity limits for the manipulator's joints. These correspond to the values specified in the universal robot description file used as basis for the Simulink controller, and are given

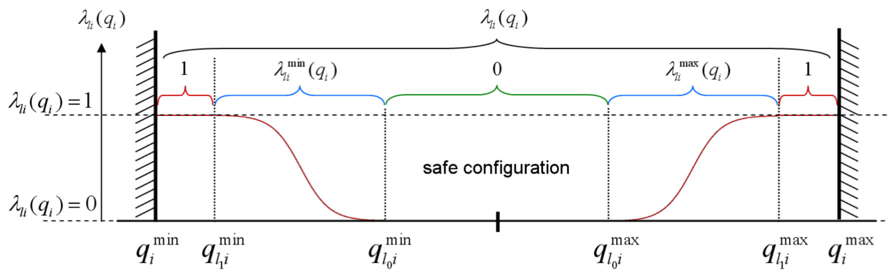


Figure 7.3.: Overview of parameters used for joint position limits avoidance, taken from [91].

in Tab. 7.2. To further ensure the safe operation of the arm an additional safety factor of 0.7 is applied.

Each joint's desired velocity command $\dot{q}_{d_i}$, calculated using Eq. 7.32, is compared to its velocity limit $\dot{q}_{max_i}$ before being dispatched to the controller. If one joint exceeds its limit, a scale factor $s = \frac{\dot{q}_{max_i}}{\dot{q}_{d_i}}$ is calculated and all joint commands are scaled according to

$$\dot{\boldsymbol{q}}_{cmd} = s\,\dot{\boldsymbol{q}}_d. \tag{7.42}$$

If multiple joints exceed their maximum, the smallest scale factor is applied to ensure all limits are being met.

i	$\dot{q}_{max_1}$	$\dot{q}_{max_2}$	$\dot{q}_{max_3}$	$\dot{q}_{max_4}$	$\dot{q}_{max_5}$	$\dot{q}_{max_6}$
$\dot{q}_{max_i}$ [°/s]	36	36	36	48	48	48

Table 7.2.: Maximum velocities $\dot{q}_{max_i}$ set for joints $i = 1 \dots 6$.

7.3. Docking Phase

Once the TCP reaches its predefined goal position and is correctly aligned with the coupling partner, we switch to a PBVS based hybrid impedance visual servoing (HIVS) scheme. This is used in conjunction with the supplied Cartesian impedance controller of the arm, to reduce the contact forces and ensure the safe and compliant contact behavior between docking interface and coupling partner.

Impedance control is a form of interaction control that enforces a desired dynamic behavior between the end-effector of the robot and the environment [92, p.29]. It is achieved by modeling the system as a dynamic impedance in the form of a mass-spring-damper system [93], where the spring stiffness and damping of the system are used to regulating the relationship between a desired input position and the resulting interaction force, see Fig. 7.4. It is especially suited for the control of tasks where contact forces between robot and environment are needed to be kept *small*, without being required to be regulated explicitly [93]. In the case of Cartesian impedance control, this relationship is given in terms of the minimal representation of the manipulator's task space coordinates characterizing the end-effector motion $\boldsymbol{x} \in \mathbb{R}^6$, introduced in Sec. 4.2.4. In our case the end-effector corresponds to the TCP.

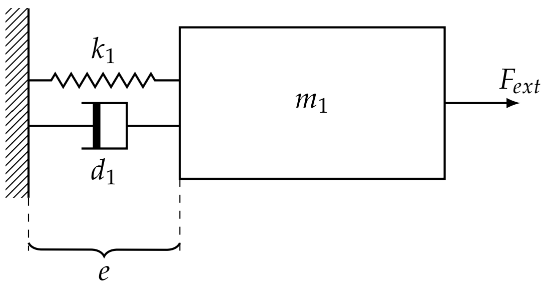


Figure 7.4.: One dimensional example of a mass-spring-damper system for impedance control. With mass m_1, damping factor d_1 and stiffness k_1 regulating the relationship between the position error $e = x - x_d$ and the force F_{ext}.

To maintain high positioning accuracy within the X-Y plane, we implement a hybrid scheme in which these DOF of the TCP frame will remain motion/position controlled while enforcing a low-impedance behavior along the Z-axis and rotational DOFs of the frame. For its description, we will begin with an overview on the implemented Cartesian impedance controller of the manipulator.

7.3.1. Cartesian Impedance Controller

To describe the control in the task space, we first need to consider the operational space formulation [94] of the dynamic robot model from Eq. 7.3. This is given by

$$\boldsymbol{M}_x(\boldsymbol{q})\ddot{\boldsymbol{x}} + \boldsymbol{C}_x(\boldsymbol{x}, \dot{\boldsymbol{x}})\dot{\boldsymbol{x}} + \boldsymbol{F}_g(\boldsymbol{x}) = \boldsymbol{\mathcal{F}}_\tau + \boldsymbol{\mathcal{F}}_{\text{ext}}, \tag{7.43}$$

where $\boldsymbol{M}_x(\boldsymbol{q})$ and $\boldsymbol{C}_x(\boldsymbol{x}, \dot{\boldsymbol{x}})$ are the matrices describing the inertia as well as Coriolis and centrifugal forces with respect to $\boldsymbol{x}$. $\boldsymbol{F}_g(\boldsymbol{x})$ is the vector of task space gravity forces and $\boldsymbol{\mathcal{F}}_\tau$, $\boldsymbol{\mathcal{F}}_{\text{ext}}$ are the vectors of forces related to the joint torque and external torques by $\boldsymbol{\tau}_\tau = \boldsymbol{J}(\boldsymbol{q})^T \boldsymbol{\mathcal{F}}_\tau$ and $\boldsymbol{\tau}_\tau = \boldsymbol{J}(\boldsymbol{q})^T \boldsymbol{\mathcal{F}}_{\text{ext}}$ [92].
By defining the error between current and desired end-effector pose $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}_d$, the dynamic relationship between $\boldsymbol{e}$ and $\boldsymbol{\mathcal{F}}_{\text{ext}}$ is given by [92, p.31]

$$\boldsymbol{\mathcal{F}}_{\text{ext}} = \boldsymbol{M}_d \ddot{\boldsymbol{e}} + \boldsymbol{D}_d \dot{\boldsymbol{e}} + \boldsymbol{K}_d \boldsymbol{e}, \tag{7.44}$$

where $\boldsymbol{M}_d$, $\boldsymbol{D}_d$ and $\boldsymbol{K}_d$ are the symmetric and positive definite matrices of the desired inertia, stiffness and damping, describing the impedance of the system. By setting the impedance to be *high*, the system can be transformed into a motion controlled one [93]. This is also possible for individual DOF of the system [93], enabling us to design a hybrid motion and impedance controlled system.

The classical impedance control law, relating position and force, is lastly derived from Eq. 7.43 and Eq. 7.44 and given by [92, p.34]

$$\begin{aligned}\boldsymbol{\mathcal{F}}_\tau = \boldsymbol{F}_g(\boldsymbol{x}) + \boldsymbol{M}_x(\boldsymbol{x})\ddot{\boldsymbol{x}}_d + \boldsymbol{C}_x(\boldsymbol{x},\dot{\boldsymbol{x}})\dot{\boldsymbol{x}} - \boldsymbol{M}_x(\boldsymbol{x})\boldsymbol{M}_d^{-1}(\boldsymbol{D}_d\dot{\boldsymbol{e}} + \boldsymbol{K}_d\boldsymbol{e}) \\ +(\boldsymbol{M}_x(\boldsymbol{x})\boldsymbol{M}_d^{-1} - \boldsymbol{I})\boldsymbol{\mathcal{F}}_{\text{ext}}.\end{aligned} \tag{7.45}$$

It requires a desired position as an input while generating a resulting force output.

In the case of the cartesian impedance controller supplied to us, explicit shaping of the desired inertia matrix $\boldsymbol{M}_d$, and thus the necessity for the feedback of external force, is being avoided by assuming that

$$\boldsymbol{M}_d = \boldsymbol{M}_x. \tag{7.46}$$

As the desired inertia of the system depends on the position $\boldsymbol{x}$ [92, p.35], Eq. 7.44 needs to be extended to consider also the relevant Coriolis and centrifugal forces [92, p.36]

$$\boldsymbol{\mathcal{F}}_{\text{ext}} = \boldsymbol{M}_d\ddot{\boldsymbol{e}} + (\boldsymbol{C}(\boldsymbol{x},\dot{\boldsymbol{x}}) + \boldsymbol{D}_d)\dot{\boldsymbol{e}} + \boldsymbol{K}_d\boldsymbol{e}. \tag{7.47}$$

Using Eq. 7.46 and Eq. 7.47 the control law given in Eq. 7.45 simplifies to [92, p.35]

$$\boldsymbol{\mathcal{F}}_\tau = \boldsymbol{F}_g(\boldsymbol{x}) + \boldsymbol{M}_x(\boldsymbol{x})\ddot{\boldsymbol{x}}_d + \boldsymbol{C}_x(\boldsymbol{x},\dot{\boldsymbol{x}})\dot{\boldsymbol{x}}_d - \boldsymbol{K}_d\boldsymbol{e} - \boldsymbol{D}_d\dot{\boldsymbol{e}} \tag{7.48}$$

and we are left with only the damping $\boldsymbol{D}_d$ and stiffness $\boldsymbol{K}_d$ term left to tune the impedance of our system.

While the choice of the values of $\boldsymbol{K}_d$ is based on the desired compliance along those DOF for the given task, the design of $\boldsymbol{D}_d$ has to account for the dynamic inertia of the system. For this purpose, the provided Cartesian impedance controller makes use of the damping design via generalized eigenvalues, discussed in [95]. It determines $\boldsymbol{D}_d$ based on the current system inertia and a constant set of damping ratios $\boldsymbol{\zeta}$ with values between $\zeta_i = 0$ for undamped, and $\zeta_i = 1$ for critically damped behavior. As the damping directly impacts the system response to oscillations, it has an important influence on the stability of the system [96].

Thus, our HIVS scheme should supply the controller with the desired values of the diagonal stiffness matrix $\boldsymbol{K}_d$, the damping factors $\boldsymbol{\zeta}$ and the desired pose of the TCP. The latter one needs to be given by a desired goal frame TCP*, described by its transform ${}^{B}T_{TCP^*}$ relative to the base frame of the manipulator.

7.3.2. Hybrid Impedance Visual Servoing

To set the manipulator into Cartesian impedance mode, the HIVS first needs to supply the controller with the desired stiffness and damping values to be applied along the axes of the TCP frame. It has to be noted, that the maximum feasible stiffness values for the controller are limited to 500 N/m for translational and 50 Nm/rad for the rotational DOF of the frame, while the damping factors are bounded between $0.0 \leq d \leq 1.0$. Since we aim at applying motion control along the X and Y axis of the TCP, to ensure it is correctly positioned throughout the whole docking procedure, these two axis are be set to the maximal available stiffness values. Thus, the diagonal matrix of the desired stiffnesses is given by

$$\boldsymbol{K}_d = \begin{bmatrix} 500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 & 0 & 0 \\ 0 & 0 & k_z & 0 & 0 & 0 \\ 0 & 0 & 0 & k_{\omega_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & k_{\omega_y} & 0 \\ 0 & 0 & 0 & 0 & 0 & k_{\omega_z} \end{bmatrix}, \tag{7.49}$$

where k_z, k_{ω_x}, k_{ω_y} and k_{ω_z} are *low* stiffness values, whose exact values will need to be determined experimentally, to ensure the correct amount of compliance along those DOF. This can be done, e.g., by carrying out several experiments emulating the docking procedure, comparing the desired and actual displacements along the end-effector axes, as well as the resulting force-position relationships. An additional interesting quantity is the impact force, quantifying the force peak on first contact between end-effector and coupling partner, as this should be kept small enough not to cause damage to the docking parts.

Similarly, the values for the damping ratios

$$\boldsymbol{\zeta} = \begin{bmatrix} \zeta_x & \zeta_y & \zeta_z & \zeta_{\omega_x} & \zeta_{\omega_y} & \zeta_{\omega_z} \end{bmatrix} \tag{7.50}$$

are best found through experimental evaluation. If they are chosen too small, we expect visible vibrations or oscillations of the system. Too large values in turn may lead to the slow responsiveness of the system to dynamic changes. Therefore, we intend to tune the values by observing the transient response of the system, particularly considering its oscillatory behavior and response time to new command inputs.

In comparison to the PBVS scheme used during the approach phase, the HIVS needs to supply the controller with the desired position in the task space. Thus, we start the description of our servoing scheme by revisiting the calculation of the desired TCP velocity ${}^{TCP}\boldsymbol{v}$ in the task space, based on the error between the current and desired TCP pose, as given in Eq. 7.14:

$${}^{TCP}\boldsymbol{v} = -\lambda \boldsymbol{L}^{-1} \boldsymbol{e}_x.$$

Instead of supplying the impedance controller with a single goal frame at the beginning of the docking phase, we periodically re-sample intermediate goal frames along the direction of the vector given by the position error. This is done to ensure the vision-based position correction is applied along the whole docking trajectory, to take into account any additional changes in the kinematic error that might be introduced by the forces acting on the end-effector during docking. Therefore, we use λ as a means to define the dynamic step size between consecutive intermediate goal frames and bind it to $\lambda_\infty = 0.1$ and $\lambda_0 = 1.0$, increasing λ as $\boldsymbol{e}$ decreases. λ_0' is considered as an additional tuning factor, influencing the number of interpolation steps along the trajectory and overall speed of the docking procedure. It determines how fast we arrive at the value for λ_0, which constitutes the final final step of the docking.

For the calculation of the next intermediate goal frame TCP^*, we start by calculating the necessary displacement $\Delta^B\boldsymbol{T}_{TCP^*}$ on the basis of the ${}^{TCP}\boldsymbol{v}$. We therefore rewrite the velocity in the form of of its twist

$$ {}^{TCP}\boldsymbol{V} = \begin{bmatrix} [\boldsymbol{\omega}]_\times & \mathbf{v} \\ \mathbf{0}_3 & 0 \end{bmatrix} \in \boldsymbol{se}(3) \tag{7.51} $$

and arrive at

$$ \Delta^B\boldsymbol{T}_{TCP^*} = {}^B\boldsymbol{T}_{TCP}{}^{TCP}\boldsymbol{V}. \tag{7.52} $$

The next goal frame to be send to the controller is then given by

$$ {}^B\boldsymbol{T}_{TCP^*} = {}^B\boldsymbol{T}_{TCP} + \frac{1}{r}\,\Delta^B\boldsymbol{T}_{TCP^*}, \tag{7.53} $$

where r is the rate, given in Hertz, at which the HIVS operates.

7.4. Experimental and Simulation Setup

All experimental evaluations of the proposed tracking and visual servoing schemes were originally intended to be conducted directly on the real rover system. However, due to the current pandemic, we expected that only limited system time will be available to us.

While the assessment of the tracking algorithm can be performed on basis of recorded ROS Bag files, the same does not hold for the tuning and testing of the closed loop visual servo control schemes.

Thus, a simulation of the rover and its environment has been implemented as part of this thesis, in order to carry out first tuning steps and experimental evaluations of the PBVS method. It serves as preparation for the final on-rover experiments, which were planned to include, inter alia, the tuning of the stiffness and damping matrices for our proposed HIVS scheme, as soon as the rover system becomes available to us. Unfortunately, in the end the rover was not available for any experiments during the time of this thesis, due to several issues introduced by substantial hardware and software updates.

Simulation Environment

The rover simulation was implemented using the Gazebo 11 Simulator [63] with ROS Melodic on a Linux OpenSuSe Leap 15 platform. It features a simplified version of the rovers environment, consisting of the LRU2 equipped with a static camera in place of the real rover's pan/tilt camera and a free floating AprilTag used as a reference for the payload docking goal frame, as can be seen in Fig. 7.5. Fig. 7.6 further shows the the perspective obtained by the simulated camera.

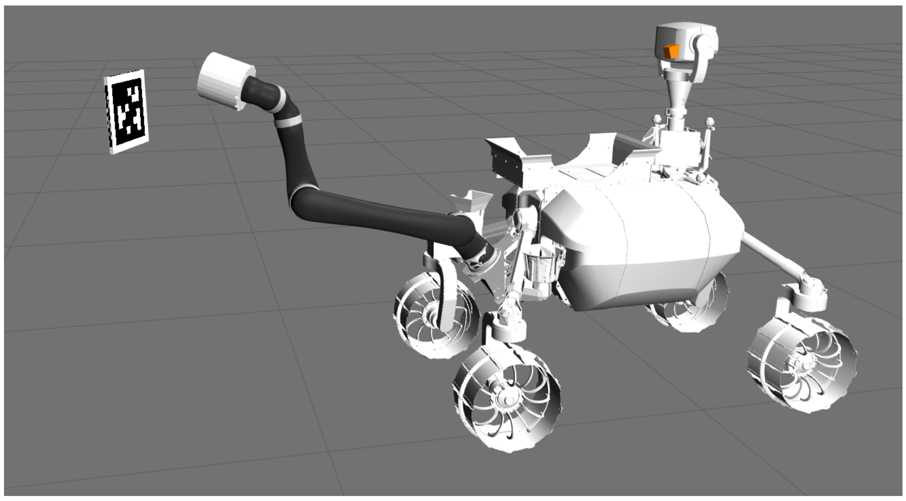


Figure 7.5.: Simulated rover scene in Gazebo. The orange cube indicates the placement of the left pan/tilt camera.

The simulated docking interface has a diameter of 10 cm. Its surface material and color where chosen to approximate those of the real interface. When modeling an object in Gazebo, several tags [97] can be used to alter the color, appearance and light dependent properties of the object. By tuning the specularity of the simulated docking interface surface, we mimic the shininess of the metallic material. The LED array is represented by a circular pattern of white spheres on the outer margin of the docking interface. Each sphere has a diameter of 7 mm. To simulate the shining of the LEDs, the emissive property [97] of the sphere surfaces was set to the maximum value.

For the detection of the AprilTag we used the apriltag_ros [98–100] package. After successful detection, the relative transform between the camera and AprilTag is added to the transformation tree and continuously updated.

We further added the Gazebo planar move plugin, to be able to easily reposition the base of the rover between experiments by specifying a desired linear (x, y) and angular z velocity of the displacement.

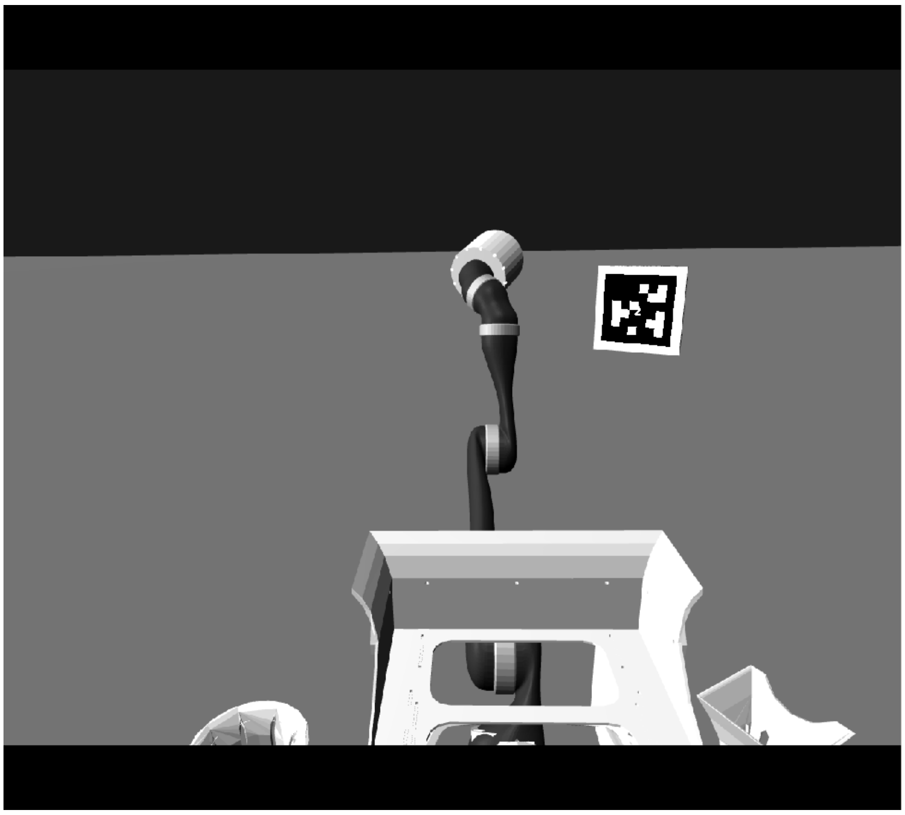


Figure 7.6.: An example image given by the image stream of the simulated camera.

For simulation and testing of the PBVS scheme, we chose to use the joint position controller provided by the ros_control [101] package. While the package also provides a joint velocity controller, we found that its current implementation shows unexpected behaviors, e.g., continued movement even when sending zero velocities, that made it inapplicable for our use case. In contrast to the controller of the real manipulator, the simulated controller thus requires joint positions as inputs instead of velocities. For this reason, we apply an additional integration step to the joint commands obtained in Eq. 7.42 using the current values of the manipulator configuration $\boldsymbol{q}(t)$ to obtain the next desired configuration

$$\boldsymbol{q}_{cmd}(t) = \boldsymbol{q}(t) + \Delta t \, \dot{\boldsymbol{q}}_{cmd}(t), \tag{7.54}$$

where $\Delta t = \frac{1}{r}$ and r is the the rate of operation of the visual servo.

7.5. Evaluation

For the simulation based evaluation of out PBVS scheme, the parameters of the tracker were altered to reliably track simulated LEDs. This is due to the lower radiance of the simulated marker spheres, which reduces their apparent size in the image stream, compared to the real LEDs. The adjusted parameters are found in Tab. 7.3

ROI size s [pixel]	**Radius r [pixel]**	**Number of lines N**	**ROI Scaling Factor rsf**
10×10	5	20	5

Table 7.3.: Values of the gradient intersect tracker for use in simulation.

In the following experiments, we defined a threshold of 2 mm for the residual error of the translational and 0.05 rad for rotational DOF of the TCP frame. As soon as all DOF reach, or are below, their threshold values, the PBVS is considered successful and is terminated. For each of the experiments multiple trials have been conducted, for which exemplary results are shown.

7.5.1. Tuning the Adaptive Gain

Before any other experiments can be carried out, the adaptive gain, introduced in Sec. 7.2.2, has to be tuned to achieve satisfactory results of the servoing scheme. The correct adjustment of the three gain variables λ_0, λ_∞ and λ_0' directly influences both the convergence of the pose error, and the time needed to reach the goal threshold. Note that the following experiments have been conducted using the DLS version of the task Jacobian's pseudo inverse, as given in Eq. 7.21.

We begin by considering λ_∞. This corresponds to the gain for very large $||e||$. An adequate value for λ_∞ can be found by conducting several trial runs, with the end-effector being positioned far away from its goal pose. Starting from a small value, such as $\lambda_\infty = 0.01$, the gain is increased with every trial, until the tracking algorithm can not keep up with the commanded movement of the end-effector anymore [102].

Since we limit the maximum values of the commanded joint velocities, as described in Sec. 7.2.2, our tracking had no difficulties to keep up with the movement of manipulator, even when the commanded joint velocities reached their maximum allowance. However, to prevent the joint velocity limits avoidance from being active throughout the whole servoing task, we found that a value of $\lambda_\infty = 0.1$ represents a good compromise between a fast convergence speed and adherence to the given limits.

The value for λ_0', indicating the slope of the gain when the norm of the error approaches zero, was set to $\lambda_0' = 10$ and kept constant for the tuning of the last gain factor

λ_0.

λ_0 is again found by trial-and-error. For this, we conduct several trials with different values of λ_0. All trials share the same initial manipulator configuration of

$$\boldsymbol{q} = \begin{bmatrix}0.0 & 2.7 & 2.7 & 0.0 & 0.0 & 0.0\end{bmatrix}^T \tag{7.55}$$

resulting in the initial task space pose relative to the base frame

$${}^{B}\boldsymbol{x}_{TCP} = \begin{bmatrix}0.18 & 0.06 & 1.10 & 0.0 & 0.0 & 1.5708\end{bmatrix}^T. \tag{7.56}$$

The task space goal pose is given by

$${}^{B}\boldsymbol{x}_{TCP^*} = \begin{bmatrix}0.49 & -0.08 & 0.59 & 0.18 & 0.77 & 0.45\end{bmatrix}^T. \tag{7.57}$$

In both cases, the rotation given by its angle/axis representation $\boldsymbol{\omega}$.

In the following we consider three exemplary cases for the choice of λ_0. A large value of $\lambda_0 = 40$, a small value of $\lambda_0 = 1$, and an well tuned choice of $\lambda_0 = 2.5$

Fig. 7.7 illustrates the system behavior for an overly large value of $\lambda_0 = 40$. It can be seen that while the system is already close to convergence after 11.4 s, the large gain applied for small pose errors leads to oscillations around the goal pose, which, in our example, become most apparent for the rotational DOF of the end-effector. It has to be remembered that the rotation around the end-effector Z-axis is neglected, leading to a constant zero error along this DOF. The oscillations visible in the other DOF are due to the the large commanded joint velocities in the close vicinity of the goal state, induced by the overly large choice for λ_0. This keeps the system from reaching a steady state. The elements of $||\boldsymbol{e}||$ do not reach their goal threshold simultaneously, and the PBVS continues its operation until stopped by the user.

An example for the effect of choosing a too small value of λ_0 is shown in Fig. 7.8. For this experiment, we set $\lambda_0 = 1$. It is apparent that both the translational and rotational error are initially reduced. The reduction of the translational error shows a linear behavior for most of the time, before flattening around 35 s after the start of the servo. The normed rotational error encounters its minimum at approximately 38 s, inducing a second, short period of linear reduction in the pose error. The reason for this is the dependency of $\lambda(||\boldsymbol{e}||)$ on the norm of the remaining pose error. A reduction in $||\boldsymbol{e}||$ hereby leads to a shift of $\lambda(||\boldsymbol{e}||)$ towards λ_0. However, due to the poor choice of λ_0, complete convergence is never reached and the trial is terminated after $t = 145$ s. At the time of termination, the error is below its threshold of 0.05 rad for all rotational DOF of the end-effector. At the same time, the residual error of its first two translational DOF, which are critical for the precise docking, remains above the allowed limit of 2 mm, with $\Delta x = 5$ mm and $\Delta y = 11$ mm, respectively.

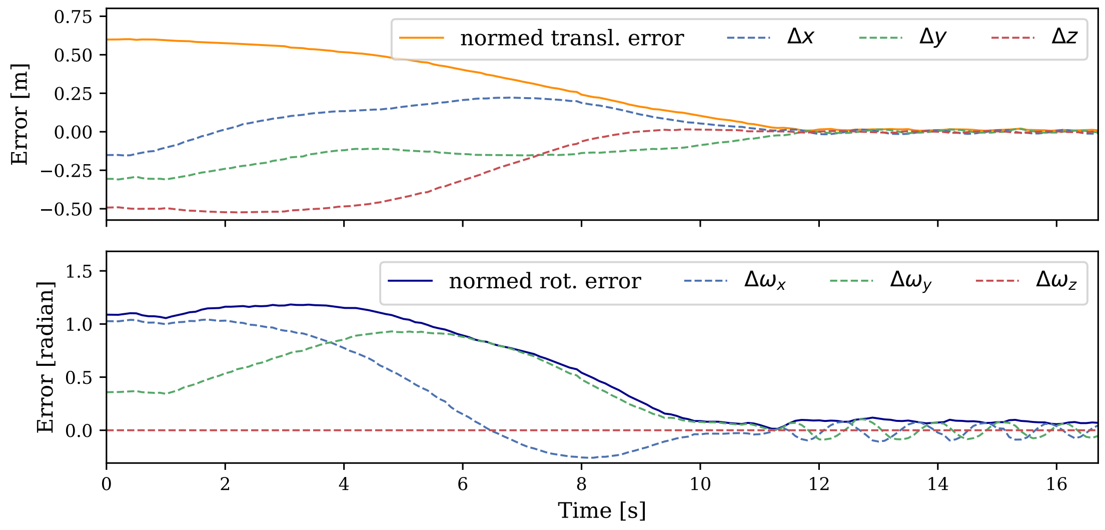


Figure 7.7.: Example of the pose error convergence for an adaptive gain with gain factors $\lambda_0 = 40$, $\lambda_\infty = 0.1$ and $\lambda_0' = 10$.

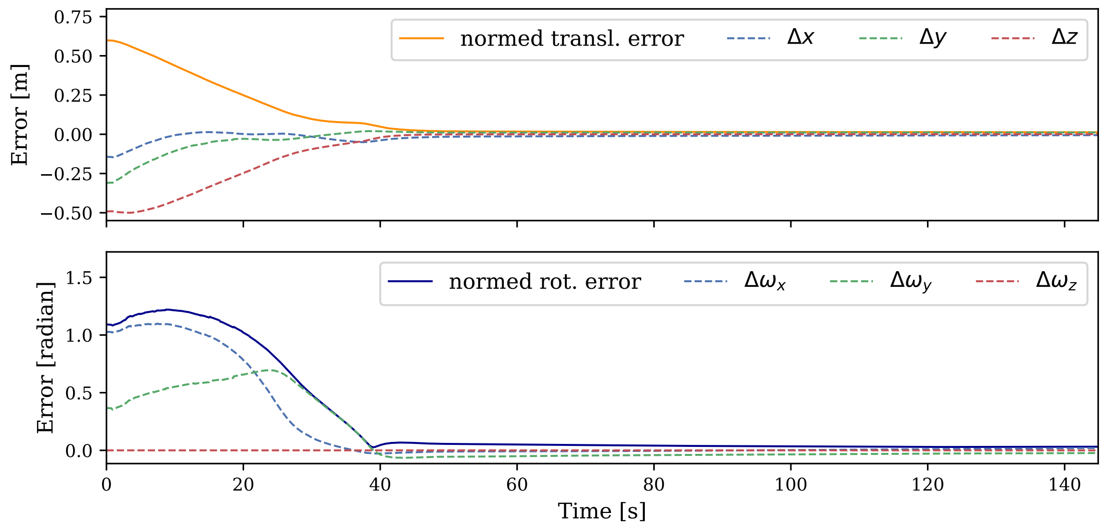


Figure 7.8.: Example of the pose error convergence for an adaptive gain with gain factors $\lambda_0 = 1$, $\lambda_\infty = 0.1$ and $\lambda_0' = 10$. The experiment was stopped after $t = 145$s, due to large residual error.

Finally, Fig. 7.9 shows the evolution of the pose error for a well tuned value of $\lambda_0 = 2.5$, which leads to convergence, while avoiding oscillations of the end-effector around the goal pose. It can be seen, that the position error decreases approximately linearly over

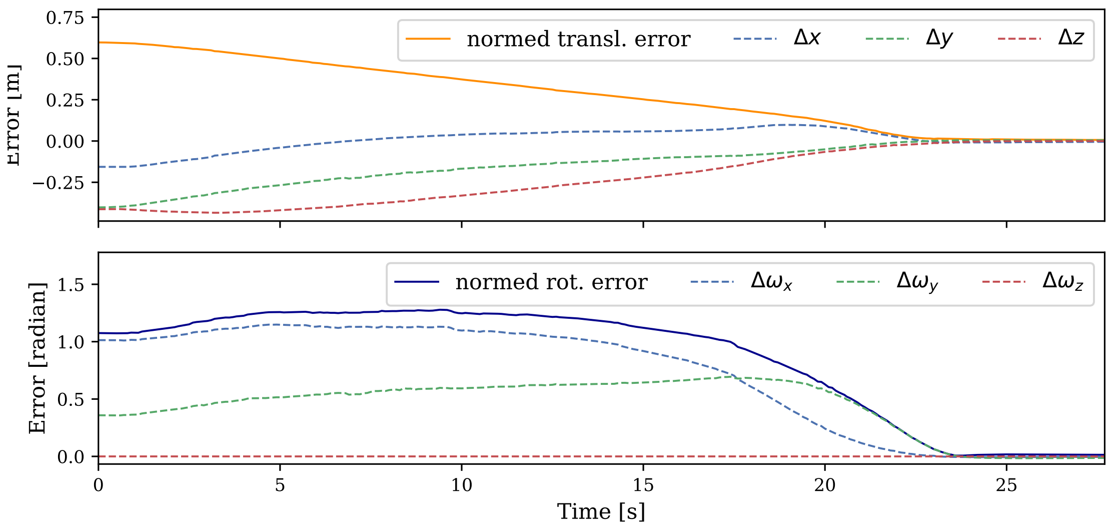


Figure 7.9.: Example of the pose error convergence for an adaptive gain with gain factors $\lambda_0 = 2.5$, $\lambda_\infty = 0.1$ and $\lambda_0' = 10$.

the course of 23 s and reaches its goal threshold after 27 s. As for the last example, the change in inclination at approximately 22 s after start again stems from the beginning shift towards λ_0. While the rotation error initially increases as the end-effector moves towards its goal position, it lastly reaches its threshold value 24 s after the start of the visual servo and remains within the threshold range until the position error reaches it, too. All following experiments are thus carried out using the composition of the gain factors: $\lambda_0 = 2.5$, $\lambda_\infty = 0.1$ and $\lambda_0' = 10$.

7.5.2. Singularity Avoidance

We proceed to compare the behavior of our visual servo in the vicinity of singularities, depending on the choice of inversion method for the task Jacobian. Therefore, we conduct two experiments, in which we consider the resulting joint velocity command issued by the visual servo in relation to the determinant $\det(\boldsymbol{J}_e)$ of the task Jacobian. In both trials we again start from the same initial configuration

$$\boldsymbol{q} = \begin{bmatrix} 0.0 & 2.7 & 2.7 & 0.0 & 0.0 & 0.0 \end{bmatrix}^T, \tag{7.58}$$

as it conveniently corresponds to a boundary singularity of the fully stretched arm. Both experiments are assigned the same end-effector goal pose relative to the detected AprilTag. Recall that a singular configuration is marked by $\det(\boldsymbol{J}_e) = 0$.

In the first experiment we examine the development of the commanded velocities in the case of the application of the Moore-Penrose pseudo inverse, $\boldsymbol{J}_e^+$, introduced in Sec. 4.2.4. The results are shown in Fig. 7.10.

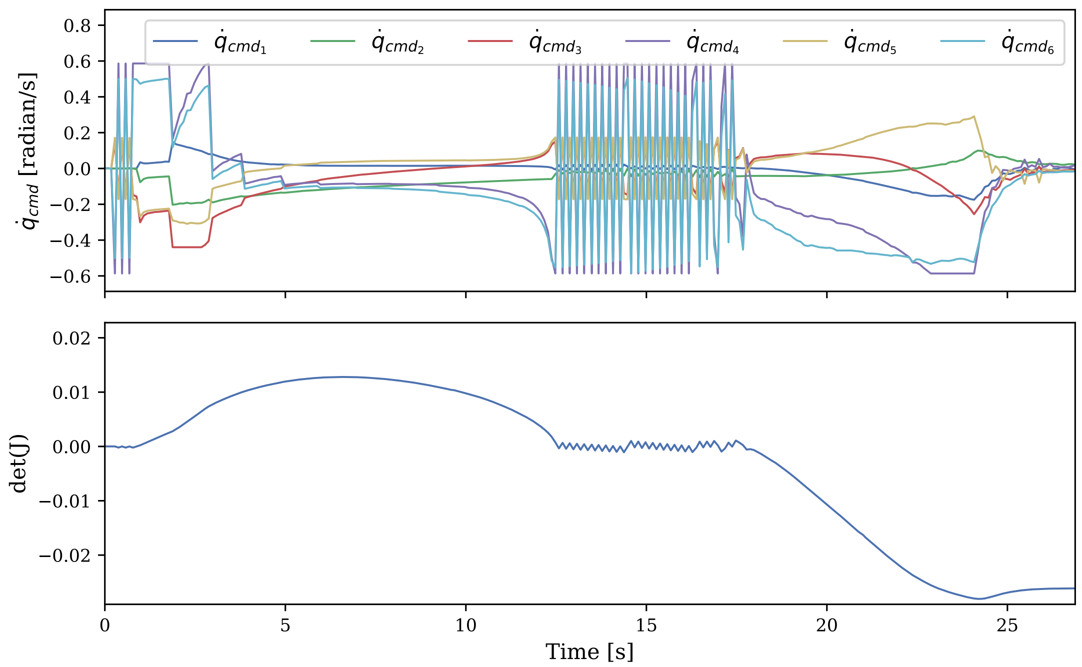


Figure 7.10.: Experiment with the standard Jacobian Pseudo-Inverse. Det($\boldsymbol{J}$) marks the determinant of the task Jacobian.

The manipulator not only starts in a singular configuration but encounters an additional wrist alignment singularity at about 12.5 s while approaching the goal pose. It can be seen in both cases that the commanded velocities of PBVS start to increase when approaching the singularity, leading to strong oscillations in the command. The highest velocity peaks can be observed for joints J_4 and J_6, as is expected for this type of singularity since the rotational axes of these joints now coincide. This manifests itself within the simulation by visible shaking and oscillation of the distal joints of the manipulator.

We point out that the visible bounds of the joint velocity oscillations are only due to the velocity limitation implemented in our PBVS scheme, and would otherwise have grown even larger. After the manipulator is stuck in immediate vicinity of the second singularity for approximately 10 s, the command signal stabilizes again and the end-effector lastly reaches its goal configuration.

In the second experiment, we instead show the system behavior for the choice of the DLS pseudo inverse $\boldsymbol{J}_e^{\#}$, presented in Sec. 7.2.2. It can be seen in Fig. 7.11 that even though the arm starts its trajectory within a singular configuration, the commanded joint velocities remain free of oscillations. When approaching the second singularity, approximately 17.5 s after the start of the visual servo, the system continues to show a stable behavior with only limited increase in the commanded joint velocities. It further passes smoothly through the singularity, without getting stuck, as was the case for the Moore-Penrose pseudo inverse.

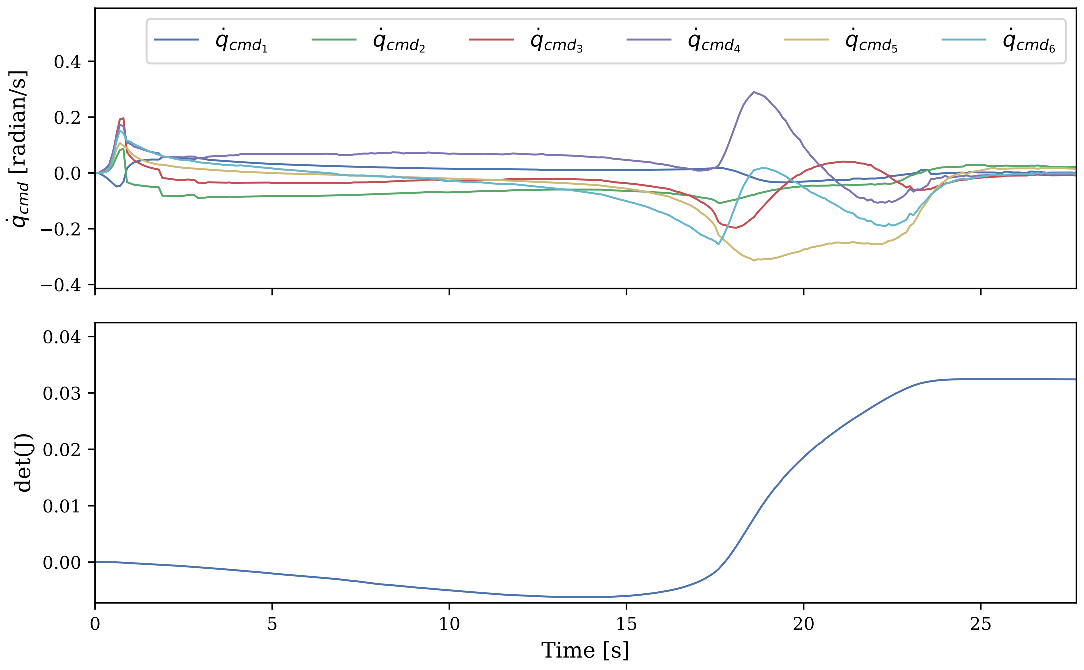


Figure 7.11.: Experiment with the damped Jabobian Pseudo-Inverse. Det($\boldsymbol{J}$) marks the determinant of the task Jacobian.

In conclusion, direct comparison of the command velocity profiles for the two tested inversion method clearly shows the superiority of the DLS over the Moore-Penrose method for the inversion of the task Jacobian for our PBVS task.

7.5.3. Joint Limits Avoidance

Lastly, we analyze the influence of the joint limits avoidance, presented in Sec. 7.2.2, on the commanded velocity signal, the induced change in position for the affected joints and the error convergence. We recall, that the joint limits avoidance is achieved by partitioning the PBVS control law into a main task $\dot{\boldsymbol{q}}_1$, whose goal is the nullification of $||\boldsymbol{e}||$, and a secondary task $\dot{\boldsymbol{q}}_2$, which aims at keeping the manipulator configuration within its joint position limits.

Since the realization of the joint limits avoidance is most important for the two non-continuous, revolute joints of our manipulator, J_2 and J_3, the focus in the following is set on the avoidance task for these joints.

We start the visual servo from an initial configuration

$$\boldsymbol{q} = \begin{bmatrix} -0.5 & 1.0 & 0.3 & -2.1 & 1.0 & -2.7 \end{bmatrix}. \tag{7.59}$$

By recalling the upper and lower hard position limits of J_2:

$$\begin{aligned} q_{2_{max}} &= 5.41\,\text{rad} \\ q_{2_{min}} &= 0.87\,\text{rad} \end{aligned}$$

and J_3:

$$\begin{aligned} q_{3_{max}} &= 5.95\,\text{rad} \\ q_{3_{min}} &= 0.33\,\text{rad} \end{aligned}$$

we see that the start configuration positions the second joint in immediate proximity to its lower position boundary, while the third joint is set even beyond the our predefined position limit.

Fig. 7.12 shows the resulting joint velocity commands $\dot{\boldsymbol{q}}_{cmd}$ in the upper plot, followed by the desired velocities commanded by the main task $\dot{\boldsymbol{q}}_1$ in the middle, and lastly shows the velocities $\dot{\boldsymbol{q}}_2$ introduced by the avoidance task on the bottom. It hereby illustrates the composition of $\dot{\boldsymbol{q}}_{cmd}$ by superposition of $\dot{\boldsymbol{q}}_1$, and $\dot{\boldsymbol{q}}_2$. The sub task thereby steers joints J_2 and J_3 away from their lower position limits. While $\dot{q}_{cmd_3}$ might not seem to correspond to the superposition of the two task velocities during the first second of operation, the reduced value of $\dot{q}_{cmd_3}$ is only due to the active joint velocity limits avoidance introduced in Sec. 7.2.2. This ensures that the addition of the sub-task command does not lead to impermissible actuator velocities.

The limits for the velocity cap are further shown in the topmost subplot of Fig. 7.13. The figure shows the relationship between the current positions of J_2 and J_3 and the resulting velocity command issued by the visual servo. The gray, horizontal lines illustrate the lower safety and soft limits of the joint positions. The vertical lines further indicate the end of the avoidance task for J_3 and J_2, respectively.

We can observe that the overstepping of the maximum position limit for q_3 leads to a to constant maximum velocity command for $\dot{q}_{cmd_3}$ during the first second of operation, due to the applied velocity limitation. This leads to a linear progress in q_3. Shortly after q_3 crosses its lower safety limit, the commanded joint velocity starts to decline linearly, until q_3 reaches its allowed region of operation after little less than 2 s. Once q_3 crosses its lower soft limit, the limit avoidance for this joint is ended, and its commanded velocity entirely depends on the main task.

A similar behavior can be observed for J_2. As q_2 is initially set within the region between soft and safety limit, the limit avoidance task starts with a velocity command of $q_2 = 2.5\,\text{rad/s}$. This decreases linearly until q_2 is approximately at the middle of the soft limit to safety limit zone. There, $\dot{q}_{cmd_2}$ changes to a constant velocity command, leading to a slower, linear convergence of q_2 towards its zone of safe operation.

By considering the evolution of the translational and rotational components of $||e||$ depicted in Fig. 7.14, we can directly recognize the impact of the avoidance task for q_3 on the normed translational error during the first two s of operation, where the increased velocity for the displacement of J_3 leads to a faster reduction of the translational error. After this, the error continues to decrease approximately linearly until reaching its goal threshold, 12.5 s after start of the PBVS.

We can conclude that the added joint position limit avoidance is successful in steering the joints away from their upper and lower position bound while respecting their velocity limitations. It thus makes a significant contribution to ensuring the safe operation of the PBVS.

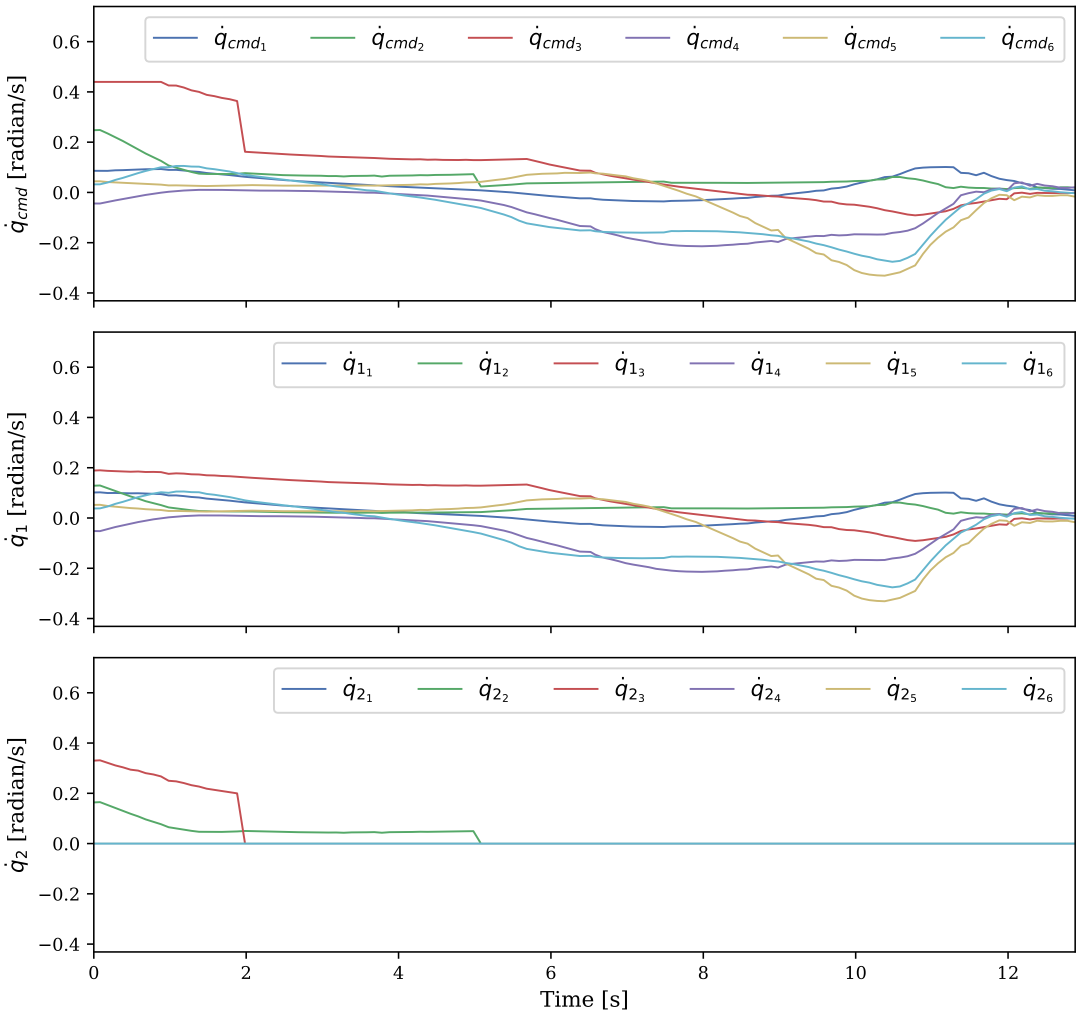


Figure 7.12.: Breakdown of the commanded joint velocities $\dot{q}_{cmd}$ (top) into those of the main task $\dot{q}_1$ (middle) and joint limit avoidance sub tasks $\dot{q}_2$ (bottom).

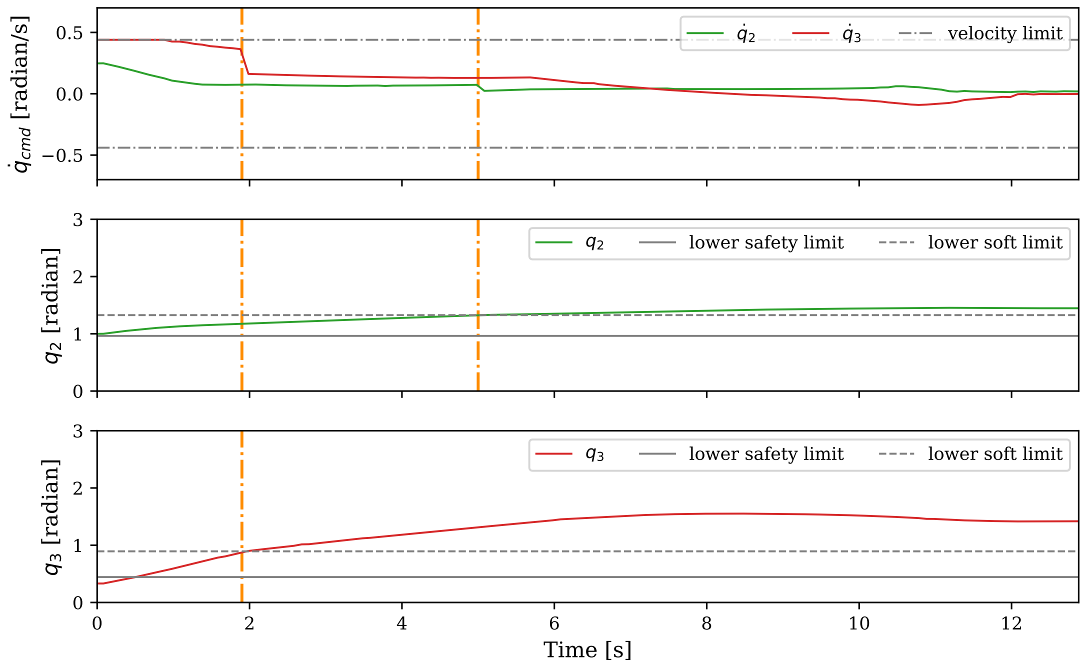


Figure 7.13.: Active velocity limit (top) and joint limit avoidance for joints J_2 (middle) and J_3 (bottom). The first vertical line from the left marks the end of the joint limit avoidance task for J_3, the second line that of J_2.

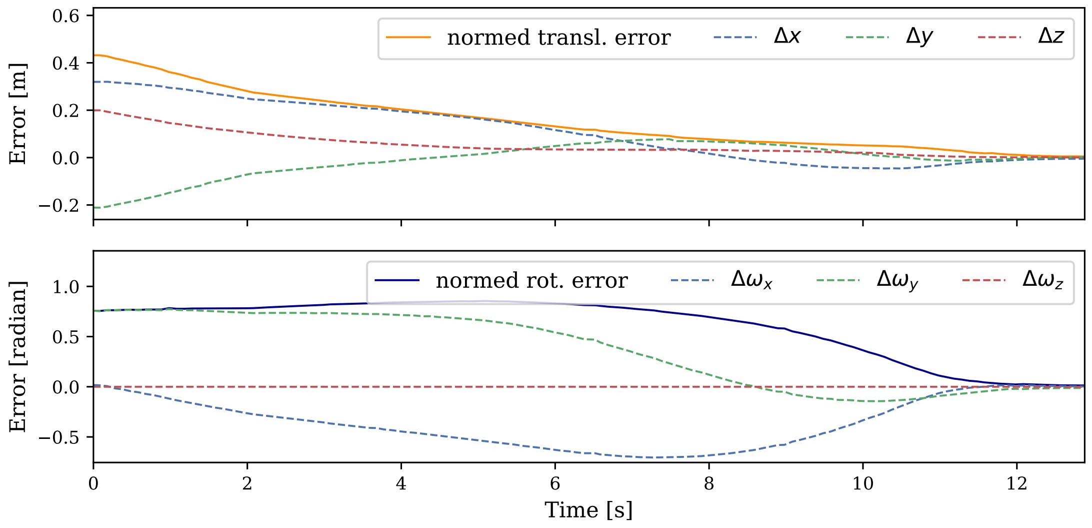


Figure 7.14.: Convergence of the error norm in the case of an active joint limit avoidance for joints J_2 and J_3.

8. Conclusion

The goal of this thesis was the design and implementation of a pipeline for robust visual servo control and tracking for the light-weight manipulator of a planetary exploration rover. The following chapter summarizes the individual steps and design choices of the presented approach. It ends with an overview on open questions that arose over the course of this thesis, as well as promising future research topics.

8.1. Summary

This thesis started with an overview on the rover components, focusing on the manipulator, its docking interface and the available data. The concept of visual servoing was introduced and a summary of related works concerning visual servoing for light-weight manipulators was given. Additionally, an overview on state-of-the-art tracking methods for visual servoing applications was presented. This was followed by the presentation of the mathematical fundamentals used throughout this work.

Thereafter, a concept for the vision-based end-effector pose correction was derived. For this purpose, the system challenges were identified and the characteristics of the manipulator's kinematic error were analyzed.

To track the array of active markers within the image stream, a gradient based tracking method was proposed, which makes use of the manipulator's kinematics as a prior to the next marker point positions. By using the MLPnP algorithm to solve for the 3D pose of the marker array, the covariance information for the retrieved image points was propagated to obtain an estimate of the 3D pose covariance.

Due to fluctuations in the accuracy of the retrieved image point locations, the visual pose estimates were afflicted by noise. Especially large noise in the retrieved rotation estimates, compared to their kinematic measurements, motivated the joint use of the vision estimates for the correction of the end-effector position and the manipulator's kinematic for the retrieval of its orientation. To reduce the noise along the translational DOF of the end-effector, the propagated uncertainty information was used to apply an ES-EKF. The filter was designed to explicitly model the positional error between the kinematic measurements and the vision-based correction as part of the state.

To serve as a basis for visual servo control, the vision pipeline must demonstrate robustness to outliers in the vision estimates. Therefore, a validation gate was included in the corresponding measurement update step of the ES-EKF, defining a region of acceptance for new measurements and rejecting potential 3D pose outliers.

To increase the manipulator's precision during the docking procedure for the collection of payload boxes, two visual servoing methods have been implemented. For movements in free space a joint limit aware PBVS scheme was proposed. To ensure the stability of the system in the vicinity of singularities, the DLS pseudo inverse method was applied to the inversion of the task Jacobian.

When the manipulator's end-effector comes into contact with coupling partner, both safe interaction control between the two components, and the correct positioning of the docking interface, are paramount. For this purpose a PBVS based HIVS scheme was designed for use with the available Cartesian impedance controller of the manipulator, which employs a high impedance behavior for the positioning of the end-effector in the docking plane, while ensuring its compliance along the remaining DOF.

First performance tests for the active marker tracking have been carried out on off-line data sets and in simulation, and show promising results. The visual tracking has shown to reduce the normed positional error by up to 77 %, while the ES-EKF showed considerable smoothing capabilities, especially during steady state operation.

We further showed the stable and successful execution of the presented PBVS scheme in simulation, while avoiding given joint position and velocity limits. Unfortunately, the rover was not available for on-line tests over the course of this thesis. For this reason, the parameter tuning and evaluation of the presented HIVS scheme have been left for future work.

8.2. Outlook

For application on the real LRU2 platform, several further experiments are necessary. Firstly, outdoor experiments under different weather and lighting conditions are indicated, in order to further evaluate the robustness of the proposed tracking approach. Even though its performance was tested for different amounts of back- and direct light in an indoor environment, this is not the same as testing outdoors, due to differing illumination properties of the light sources.

Furthermore, the ES-EKF needs to be tuned for use with the real rover, as it was so far only possible to test its operability in simulation. Especially its process noise covariance matrix has to be set in accordance to the desired smoothing-to-dynamics trade-off. It should be noted, that the initial choice of using the ES-EKF was, inter alia, based on its good applicability to constrained quantities such as quaternions [103]. Thus, allowing for the easy integration of the 3D rotation error as part of the state. However, due to the strong noise in the rotation estimates returned by MLPnP, only the position of the marker pose could be corrected so far. In the future, the amount of noise could be reduced, e.g., by adding more marker points to the active array or by exchanging the current planar marker for a three dimensional one, thus allowing for an improved MLPnP performance.

As far as the presented PBVS scheme is concerned, additional fine tuning might become necessary for application on the rover, whereas the joint limits have already been chosen in correspondence to those of the real platform. An inconvenience that was occasionally encountered during the simulation experiments, depending on the start and goal point of the movement, was the blockage of the cameras optical axis by parts of the manipulator, leading to partial or full occlusion of the marker array. In this context, especially the third joint of the manipulator lead to blockages of the view. While this is negligible in cases when the manipulator is still far away from its docking partner, the visibility of the end-effector is of high importance when coming into close proximity to the docking target, as otherwise its position can not be reliably corrected. A solution to this problem could be found by defining the line of sight between the camera and marker array as a virtual, conic or cylindrical obstacle and introducing an additional sub task into the servoing function. This sub task would then generate the necessary obstacle avoidance command, e.g., based on the the well known concept of the artificial potential field method [104], to keep the relevant joints from blocking the view of the LED array.

Lastly, unavailability of the rover due to restricted institute access and several issues that emerged after major hardware and software updates of the platform, did not allow for any testing of the proposed HIVS scheme. Both the stiffness matrix and damping factors will need to be tuned, by considering the force-displacement relationship, stability and responsiveness of the system.

A. General Addenda

A.1. Gradient Intersect Tracking Algorithm

Algorithm 1: Gradient intersect tracking

```
Result: Refined seed point $x$ and its covariance $\Sigma$
initialization;
for seed point $x$ in seed points do
    select seed ROI;
    calculate gradient field and magnitude;
    for $i \leq max\_iterations$ do
        sample $N$ lines through $x$;
        find point with strongest gradient $g_{max}$ in a radius $r$ along each line;
        initialize an empty intersection image $M_{inter}$;
        for $g_i$ in $g_{max}$ do
            create a line for $g_i$ in $M_{inter}$;
        end
        set all entries where $M_{inter}(x, y) == 1$ to 0;
        calculate central moments of $M_{inter}$;
        calculate the centroid $x'$ of $M_{inter}$;
        if $i == max\_iterations$ then
            calculate covariance $\Sigma'$ of $x'$;
            $\Sigma \leftarrow \Sigma'$ ;
        end
        $x \leftarrow x'$ ;
    end
end
```

List of Figures

List of Tables

Acronyms

CAD computer-aided design.

DLR German Aerospace Center.

DLS damped least-squares.

DOF degrees of freedom.

EKF Extended Kalman filter.

ES-EKF Error-State Extended Kalman filter.

GIT gradient intersect tracking.

HIVS hybrid impedance visual servoing.

IBVS image-based visual servoing.

LED light-emitting diode.

LOWESS locally weighted scatterplot smoothing.

LRU2 Lightweight Rover Unit 2.

MLPnP maximum likelihood perspective-n-point.

PBVS position-based visual servoing.

PnP perspective-n-point.

ROBEX Robotic Exploration of Extreme Environments.

ROI region of interest.

ROS robot operating system.

TCP tool center point.

Bibliography

[1] M. J. Schuster, M. G. Müller, S. G. Brunner, H. Lehner, P. Lehner, A. Dömel, M. Vayugundla, F. Steidle, P. Lutz, R. Sakagami, L. Meyer, R. Belder, M. Smíšek, W. Stürzl, R. Triebel, and A. Wedler, "Towards heterogeneous robotic teams for collaborative scientific sampling in lunar and planetary environments", in *Workshop on Informed Scientific Sampling in Large-scale Outdoor Environments at the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[2] P. Lehner, S. Brunner, A. Dömel, H. Gmeiner, S. Riedel, B. Vodermayer, and A. Wedler, "Mobile manipulation for planetary exploration", in *IEEE Aerospace Conference*, IEEE, 2018, pp. 1–11.

[3] *Robex project webpage*, Accessed: 31.03.2021. [Online]. Available: `http://www.robex-allianz.de`.

[4] A. Wedler, M. Vayugundla, H. Lehner, P. Lehner, M. J. Schuster, S. G. Brunner, W. Stürzl, A. Dömel, H. Gmeiner, B. Vodermayer, B. Rebele, I. L. Grixa, K. Bussmann, J. Reill, B. Willberg, A. Maier, P. Meusel, F. Steidle, M. Smíšek, M. Hellerer, M. Knapmeyer, F. Sohl, A. Heffels, L. Witte, C. Lange, R. Rosta, N. Toth, S. Völk, A. Kimpe, P. Kyr, and M. Wilde, "First results of the ROBEX analogue mission campaign: Robotic deployment of seismic networks for future lunar missions", in *Proceedings of the International Astronautical Congress*, International Astronautical Federation (IAF), vol. 68, 2017.

[5] L. Meyer, K. H. Strobl, and R. Triebel, "Robust vision-based pose correction for a robotic manipulator using active markers", in *International Symposium on Experimental Robotics*, 2021.

[6] N. Vahrenkamp, S. Wieland, P. Azad, D. Gonzalez, T. Asfour, and R. Dillmann, "Visual servoing for humanoid grasping and manipulation tasks", in *Proceedings of the IEEE/RAS International Conference on Humanoid Robots*, IEEE, 2008, pp. 406–412.

[7] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control", *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, 1996.

[8] M. J. Schuster, C. Brand, S. Brunner, P. Lehner, J. Reill, S. Riedel, T. Bodenmüller, K. Bussmann, S. Büttner, A. Dömel, W. Friedl, I. Grixa, M. Hellerer, H. Hirschmüller, M. Kassecker, Z. C. Marton, C. Nissler, F. Ruess, M. Suppa, and A. Wedler, "The LRU rover for autonomous planetary exploration and its success in the SpaceBotCamp challenge", in *Proceedings of the International Conference on Autonomous Robot Systems and Competitions*, IEEE, 2016, pp. 7–14.

[9] *Kinova^{TM} ultra lightweight robotic arm - 6 DoF - specifications*, Accessed: 18.02.2021. [Online]. Available: `https://www.kinovarobotics.com/sites/default/files/ULWS-RA-JAC-6D-SP-INT-EN%5C%20201804-1.2%5C%20%5C%28KINOVA%E2%84%A2%5C%20Ultra%5C%20lightweight%5C%20robotic%5C%20arm%5C%206%5C%20DOF%5C%20Specifications%5C%29.pdf`.

[10] A. Campeau-Lecours, H. Lamontagne, S. Latour, P. Fauteux, V. Maheu, F. Boucher, C. Deguire, and L.-J. L'Ecuyer, "Kinova modular robot arms for service robotics applications: Concepts, methodologies, tools, and applications", in *Rapid Automation*. Jan. 2019, pp. 693–719.

[11] T. Foote, "Tf: The transform library", in *Proceedings of the IEEE International Conference on Technologies for Practical Robot Applications*, ser. Open-Source Software workshop, Apr. 2013, pp. 1–6.

[12] D. N. Bhat and S. K. Nayar, "Stereo and specular reflection", *International Journal of Computer Vision*, vol. 26, no. 2, pp. 91–106, 1998.

[13] M. J. Schuster, M. G. Müller, S. G. Brunner, H. Lehner, P. Lehner, R. Sakagami, A. Dömel, L. Meyer, B. Vodermayer, R. Giubilato, M. Vayugundla, J. Reill, F. Steidle, I. von Bargen, K. Bussmann, R. Belder, P. Lutz, W. Stürzl, M. Smíšek, M. Maier, S. Stoneman, A. F. Prince, B. Rebele, M. Durner, E. Staudinger, S. Zhang, R. Pöhlmann, E. Bischoff, C. Braun, S. Schröder, E. Dietz, S. Frohmann, A. Börner, H. Hübers, B. Foing, R. Triebel, A. O. Albu-Schäffer, and A. Wedler, "The ARCHES space-analogue demonstration mission: Towards heterogeneous teams of autonomous robots for collaborative scientific sampling in planetary exploration", *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5315–5322, 2020.

[14] A. Krupa, J. Gangloff, C. Doignon, M. F. De Mathelin, G. Morel, J. Leroy, L. Soler, and J. Marescaux, "Autonomous 3D positioning of surgical instruments in robotized laparoscopic surgery using visual servoing", *IEEE Transactions on Robotics and Automation*, vol. 19, no. 5, pp. 842–853, 2003.

[15] S. S. Mehtatt, W. E. Dixon, D. Mac Arthur, and C. D. Crane, "Visual servo control of an unmanned ground vehicle via a moving airborne monocular camera", in *American Control Conference*, 2006, pp. 6–16.

[16] J. García, A. Rodríguez, J. Estremera, B. Santamaria, D. Gonzalez, and M. Armendia, "Visual servoing and impedance control in robotic manipulators for on-orbit servicing", in *Proceedings of the International Conference on Emerging Technologies and Factory Automation*, IEEE, vol. 1, 2020, pp. 734–741.

[17] J. Hill, "Real-time control of a robot with a mobile camera", in *Proceedings of the International Symposium on Industrial Robots*, 1979, pp. 233–246.

[18] F. Chaumette, S. Hutchinson, and P. Corke, "Visual servoing", in *Springer Handbook of Robotics*, Springer, 2016, pp. 841–866.

[19] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing", in *The confluence of vision and control*, Springer, 1998, pp. 66–78.

[20] A. Sanderson, "Image-based visual servo control using relational graph error signal", in *Proceedings of the International Conference on Cybernetics and Society*, 1980.

[21] D. F. DeMenthon and L. S. Davis, "Model-based object pose in 25 lines of code", *International Journal of Computer Vision*, vol. 15, no. 1-2, pp. 123–141, 1995.

[22] S. Urban, J. Leitloff, and S. Hinz, "MLPnP - A real-time maximum likelihood solution to the Perspective-n-Point problem", in *Proceedings of ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 3, 2016, pp. 131–138.

[23] G. Chesi and K. Hashimoto, "A simple technique for improving camera displacement estimation in eye-in-hand visual servoing", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1239–1242, 2004.

[24] F. Dornaika and C. García, "Pose estimation using point and line correspondences", *Real-Time Imaging*, vol. 5, no. 3, pp. 215–230, 1999.

[25] J.-P. Tarel and D. B. Cooper, "The complex representation of algebraic curves and its simple exploitation for pose estimation and invariant recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 7, pp. 663–674, 2000.

[26] É. Marchand and F. Chaumette, "Virtual visual servoing: A framework for real-time augmented reality", *Computer Graphics Forum*, vol. 21, no. 3, pp. 289–297, 2002.

[27] E. Malis, F. Chaumette, and S. Boudet, "2 1/2 D visual servoing", *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 238–250, 1999.

[28] P. Ardón, M. Dragone, and M. S. Erden, "Reaching and grasping of objects by humanoid robots through visual servoing", in *International Conference on Human Haptic Sensing and Touch Enabled Computer Applications*, Springer, 2018, pp. 353–365.

[29] V. Lippiello, B. Siciliano, and L. Villani, "A position-based visual impedance control for robot manipulators", in *Proceedings of the IEEE International Conference on Robotics and Automation*, IEEE, 2007, pp. 2068–2073.

[30] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision", in *Proceedings of the International Joint Conference on Artificial Intelligence*, vol. 2, Vancouver, BC, Canada: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679.

[31] M. Haag and H.-H. Nagel, "Combination of edge element and optical flow estimates for 3D-model-based vehicle tracking in traffic image sequences", *International Journal of Computer Vision*, vol. 35, no. 3, pp. 295–319, 1999.

[32] A. Comport, É. Marchand, and F. Chaumette, "Robust model-based tracking for robot vision", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 2004, pp. 692–697.

[33] V. Lepetit and P. Fua, *Monocular model-based 3D tracking of rigid objects*. Now Publishers Inc, 2005.

[34] R. Muñoz-Salinas, M. J. Marıén-Jimenez, E. Yeguas-Bolivar, and R. Medina-Carnicer, "Mapping and localization from planar markers", *Pattern Recognition*, vol. 73, pp. 158–171, 2018.

[35] E. Olson, "Apriltag: A robust and flexible visual fiducial system", in *Proceedings of the IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 3400–3407.

[36] K. Dorfmüller and H. Wirth, "Real-time hand and head tracking for virtual environments using infrared beacons", in *Proceedings of the International Workshop on Capture Techniques for Virtual Environments*, Springer, 1998, pp. 113–127.

[37] P. Huber, *Robust Statistics*, ser. Wiley Series in Probability and Statistics - Applied Probability and Statistics Section Series. Wiley Online Library, 2004.

[38] P. Bouthemy, "A maximum likelihood framework for determining moving edges", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 5, pp. 499–511, 1989.

[39] J. Shi and C. Tomasi, "Good features to track", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 1994, pp. 593–600.

[40] B.-K. Seo, H. Park, J.-I. Park, S. Hinterstoisser, and S. Ilic, "Optimal local searching for fast and robust textureless 3D object tracking in highly cluttered backgrounds", *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, pp. 99–110, Jan. 2014.

[41] E. Trucco and K. Plakas, "Video tracking: A concise survey", *IEEE Journal of Oceanic Engineering*, vol. 31, no. 2, pp. 520–529, 2006.

[42] É. Marchand and F. Chaumette, "Feature tracking for visual servoing purposes", *Robotics and Autonomous Systems*, vol. 52, no. 1, pp. 53–70, 2005.

[43] É. Marchand, F. Spindler, and F. Chaumette, "ViSP for visual servoing: a generic software platform with a wide class of robot control skills", *IEEE Robotics and Automation Magazine*, vol. 12, no. 4, pp. 40–52, 2005.

[44] T. Krajník, M. Nitsche, J. Faigl, P. Vaněk, M. Saska, L. Přeučil, T. Duckett, and M. Mejail, "A practical multirobot localization system", *Journal of Intelligent and Robotic Systems*, vol. 76, no. 3, pp. 539–562, 2014.

[45] L. C. Mak and T. Furukawa, "A 6 DoF visual tracking system for a miniature helicopter", in *Proceedings of the International Conference on Sensing Technology*, 2007, pp. 32–37.

[46] R. Szeliski, *Computer vision: algorithms and applications*. Springer, 2010.

[47] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[48] Z. Zhang, "A flexible new technique for camera calibration", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1330–1334, Dec. 2000.

[49] *OpenCV: Camera calibration and 3D reconstruction*, Accessed:21.04.2021. [Online]. Available: `https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html`.

[50] D. C. Brown, "Close-range camera calibration", *Photogrammetric Engineering*, vol. 37, no. 8, pp. 855–866, 1971.

[51] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Springer, 2016.

[52] K. M. Lynch and F. C. Park, *Modern Robotics*. Cambridge University Press, 2017.

[53] J. Schmidt and H. Niemann, "Using quaternions for parametrizing 3D rotations in unconstrained nonlinear optimization", in *Proceedings of the Vision Modeling and Visualization Conference*, vol. 1, 2001, pp. 399–406.

[54] M. D. Shuster *et al.*, "A survey of attitude representations", *Navigation*, vol. 8, no. 9, pp. 439–517, 1993.

[55] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer, 2010.

[56] V. Usenko, "Visual-inertial navigation for autonomous vehicles", Ph.D. dissertation, Technical University of Munich, 2019.

[57] L. Ferraz Colomina, X. Binefa, and F. Moreno-Noguer, "Leveraging feature uncertainty in the PnP problem", in *Proceedings of the British Machine Vision Conference*, 2014, pp. 1–13.

[58] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods", *IEEE Journal of Robotics and Automation*, vol. 17, no. 1-19, p. 16, 2004.

[59] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley Online Library, 2006.

[60] Y. Pawitan, *In all likelihood: statistical modelling and inference using likelihood*. Oxford University Press, 2001.

[61] K. Madsen, H. Nielsen, and O. Tingleff, "Methods for non-linear least squares problems", University Lecture, 2004.

[62] S. Popić and B. Miloradović, "Light weight robot arms-an overview", *Infoteh-Jahorina*, vol. 14, 2015.

[63] *Gazebo simulator*, Accessed: 02.01.2021. [Online]. Available: `http://gazebosim.org`.

[64] D. Jacobs, *Image gradients*, Class Notes for CMSC 426, Accessed: 03.04.2021, 2005. [Online]. Available: `http://www.cs.umd.edu/~djacobs/CMSC426/ImageGradients.pdf`.

[65] R. Mukundan and K. Ramakrishnan, *Moment functions in image analysis: theory and applications*. World Scientific, 1998.

[66] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography", *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[67] L. Quan and Z. Lan, "Linear n-point camera pose determination", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 774–780, 1999.

[68] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate O(n) solution to the PnP problem", *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009.

[69] L. Kneip, H. Li, and Y. Seo, "UPnP: An optimal O(n) solution to the absolute pose problem with universal applicability", in *European Conference on Computer Vision*, Springer, 2014, pp. 127–142.

[70] W. Förstner, "Minimal representations for uncertainty and estimation in projective spaces", in *Asian Conference on Computer Vision*, Springer, 2010, pp. 619–632.

[71] G. Bradski, "The OpenCV Library", *Dr. Dobb's Journal of Software Tools*, 2000.

[72] P. Corke, *Robotics, vision and control: fundamental algorithms in MATLAB®*, 2nd ed. Springer, 2017, vol. 118.

[73] Y. Kim and H. Bang, "Introduction and implementations of the Kalman filter", *IntechOpen*, pp. 1–16, 2019.

[74] A. Becker, *Multidimensional Kalman filter - summary*, Accessed: 22.04.2021. [Online]. Available: `https://www.kalmanfilter.net/multiSummary.html`.

[75] F. Govaers, *Introduction and Implementations of the Kalman Filter*. IntechOpen, 2019.

[76] F. L. Markley, "Multiplicative vs. additive filtering for spacecraft attitude determination", *Dynamics and Control of Systems and Structures in Space*, no. 467-474, p. 48, 2004.

[77] L. Markley, "Attitude error representations for Kalman filtering", *Journal of Guidance Control and Dynamics*, vol. 26, pp. 311–317, Mar. 2003.

[78] B. Ekstrand, "Some aspects on filter design for target tracking", *Journal of Control Science and Engineering*, 2012.

[79] J. L. Blanco Claraco, "A tutorial on SE(3) transformation parameterizations and on-manifold optimization", ETS Ingeniería Informática Universidad de Málaga, Tech. Rep. 012010, May 2020. [Online]. Available: `https://ingmec.ual.es/~jlblanco/papers/jlblanco2010geometry3D_techrep.pdf`.

[80] J. Solà, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics", 2018. arXiv: `1812.01537 [cs.RO]`.

[81] A. Crivellaro, M. Rad, Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit, "Robust 3D object tracking from monocular images using stable parts", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1465–1479, 2017.

[82] G. Chang, "Robust Kalman filtering based on Mahalanobis distance as outlier judging criterion", *Journal of Geodesy*, vol. 88, no. 4, pp. 391–401, 2014.

[83] G. Plett, *Episode 3.3.5: Can we automatically detect bad measurements with a Kalman filter? [mooc lecture]*, Accessed: 20.04.2021, 2018. [Online]. Available: `https://www.coursera.org/lecture/battery-state-of-charge/3-3-5-can-we-automatically-detect-bad-measurements-with-a-kalman-filter-tc7ce`.

[84] *Table: Chi-square probabilities*, Accessed: 21.04.2021. [Online]. Available: `https://people.richland.edu/james/lecture/m170/tbl-chi.html`.

[85] J. Moreno–Valenzuela and V. Santibáñez, "Robust saturated PI joint velocity control for robot manipulators", *Asian Journal of Control*, vol. 15, no. 1, pp. 64–79, 2013.

[86] F. Chaumette and S. Hutchinson, "Visual servo control. I. basic approaches", *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[87] F. Chaumette and S. Hutchinson, "Visual servo control. II. advanced approaches [tutorial]", *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007.

[88] *Kinova gen2 ultra lightweight robot user guide*, Accessed: 21.03.2021, 2019. [Online]. Available: `https://www.kinovarobotics.com/sites/default/files/UG-009_KINOVA_Gen2_Ultra_lightweight_robot_User_guide_EN_R02.pdf`.

[89] N. Mansard and F. Chaumette, "Task sequencing for high-level sensor-based control", *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 60–72, 2007.

[90] A. Liegeois *et al.*, "Automatic supervisory control of the configuration and behavior of multibody mechanisms", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7, no. 12, pp. 868–871, 1977.

[91] M. Marey and F. Chaumette, "New strategies for avoiding robot joint limits: Application to visual servoing using a large projection operator", in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2010, pp. 6222–6227.

[92] C. Ott, *Cartesian impedance control of redundant and flexible-joint robots*. Springer, 2008.

[93] A. De Luca, *Impedance control [lecture]*, Accessed: 01.03.2021, 2020. [Online]. Available: `http://www.diag.uniroma1.it/deluca/rob2_en/15_ImpedanceControl.pdf`.

[94] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation", *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[95] A. Albu-Schaffer, C. Ott, U. Frese, and G. Hirzinger, "Cartesian impedance control of redundant robots: Recent results with the DLR-light-weight-arms", in *Proceedings of the IEEE International Conference on Robotics and Automation*, IEEE, vol. 3, 2003, pp. 3704–3709.

[96] D. Trumper and S. Dubowsky, *Modeling dynamics and control I [lecture]*, Accessed: 01.05.2021. License: Creative Commons BY-NC-SA, 2005. [Online]. Available: `https://ocw.mit.edu/courses/mechanical-engineering/2-003-modeling-dynamics-and-control-i-spring-2005/readings/notesinstalment2.pdf`.

[97] *Gazebo: Color and texture models [tutorial]*, Accessed: 05.01.2021. [Online]. Available: `http://gazebosim.org/tutorials?tut=color_model`.

[98] D. Malyuta, C. Brommer, D. Hentzen, T. Stastny, R. Siegwart, and R. Brockers, "Long-duration fully autonomous operation of rotorcraft unmanned aerial systems for remote-sensing data acquisition", *Journal of Field Robotics*, 2019.

[99] C. Brommer, D. Malyuta, D. Hentzen, and R. Brockers, "Long-duration autonomy for small rotorcraft UAS including recharging", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2018.

[100] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2016, pp. 4193–4198.

[101] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. Rodríguez Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, and E. Fernández Perdomo, "Ros_control: A generic and simple control framework for ROS", *The Journal of Open Source Software*, 2017.

[102] *Visual servoing platform (visp): How to boost your visual servo control law [tutorial]*, Accessed: 01.05.2021. [Online]. Available: `https://visp-doc.inria.fr/doxygen/visp-daily/tutorial-boost-vs.html#adaptive_gain_tune`.

[103] J. Solà, *Quaternion kinematics for the error-state Kalman filter*, 2017. arXiv: `1711.02508 [cs.RO]`.

[104] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots", in *Autonomous robot vehicles*, Springer, 1986, pp. 396–404.