POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Dynamics-Consistent Trajectory Mapping and Emulation in the RAFFAELLO Robotic Testbed: Methods with Digital Twin Validation and Dataset Generation

Tesi di Laurea Magistrale in
Space Engineering - Ingegneria Spaziale

Author: **Stefano Belletti**

Student ID: 252781
Advisor: Prof. Paolo Panicucci
Co-advisors: Alban Beshaj, Fabio Ornati
Academic Year: 2024-25

Abstract

Deep space missions typically face navigation challenges for vision-based algorithms, which need to be tested with meaningful data. Several techniques can be employed, each with a different level of abstraction: the higher the abstraction, the less representative the generated data.

To face these issues, Hardware-in-the-Loop tests have gained popularity thanks to their ability to provide high-fidelity simulations. Using a mockup of the real target and a camera mounted on high torque robotic manipulators, it is possible to simulate a real life scenario. These tests are extremely important for missions where in situ tests are prohibitive in terms of cost or simply not possible. However, these datasets must be acquired with a good level of consistency: both in terms of preservation of the trajectory constraints, and from a dynamic viewpoint. The thesis aims to produce a mapping technique which can be applied to the RAFFAELLO testbed: a procedure that maps relative orbital dynamics into the laboratory facility, ensuring both high-fidelity and dynamic-consistent results.

Robotics fundamentals are combined with nonlinear optimization, where the mapping solution is solved in a constrained environment. Digital twin methodologies are used to model the hardware of the facility, in order to verify and validate the optimization results, while Hardware-in-the-Loop tests are carried out to acquire datasets in the existing facility.

Keywords: Trajectory Mapping; Dynamic Consistency; Robotic Testbed; Digital Twin; Hardware-in-the-Loop; Dataset Generation.

Abstract in lingua italiana

Le missioni nello spazio profondo affrontano tipicamente sfide di navigazione per algoritmi vision-based, che devono essere testati con dati significativi. Diverse tecniche possono essere impiegate, ciascuna con un diverso livello di astrazione: più alto è il livello di astrazione, meno rappresentativi sono i dati generati.

Per affrontare questi problemi, i test Hardware-in-the-Loop hanno guadagnato popolarità grazie alla loro capacità di fornire simulazioni ad alto livello di fedeltà. Utilizzando un mockup del target reale e una fotocamera montata su manipolatori robotici high torque, è possibile simulare uno scenario reale. Questi test sono estremamente importanti per le missioni in cui i test in situ sono proibitivi in termini di costi o semplicemente non possibili. Tuttavia, questi dataset devono essere acquisiti con un buon livello di consistenza: sia in termini di preservazione dei vincoli di traiettoria, sia dal punto di vista dinamico. La tesi mira a produrre una tecnica di mapping che possa essere applicata al testbed RAFFAELLO: una procedura che mappa le dinamiche relative in laboratorio, garantendo sia risultati high-fidelity, sia consistenti dal punto di vista dinamico.

I fondamenti della robotica sono combinati con nonlinear optimization, dove la soluzione di mapping è risolta in un ambiente vincolato. Le metodologie di digital twin sono utilizzate per modellare l'hardware della facility, al fine di verificare e validare i risultati dell'ottimizzazione, mentre i test Hardware-in-the-Loop vengono eseguiti per acquisire dataset nella facility esistente.

Parole chiave: Trajectory Mapping; Consistenza Dinamica; Testbed Robotico; Digital Twin; Hardware-in-the-Loop; Generazione di Dataset.

Contents

1 | Literature Review

The Guidance, Navigation and Control (GNC) subsystem for missions to small celestial bodies is safety critical and in-situ tests are usually prohibitive in terms of cost. For this reason, facility tests must me performed to evaluate algorithms robustness, verification and validation. These tests are carried out using both Software-in-the-Loop (SIL) and Hardware-in-the-Loop (HIL) methodologies, since complete subsystems or smaller parts can be tested with high fidelity.

However, simulations with software modeling, such as in Digital Twinning methodologies, often introduce biases and imperfections given by the complexity of the task, this is the so called "sim-to-real" gap [4]. For this reason, many research centers prefer employing HIL testbeds, given their ability to physically reproduce parts of real life missions. These facilities are usually composed by robotic manipulators which can be useful for their ability to reproduce three-dimensional movements with many available Degrees Of Freedom (DOF).

The focus of the thesis is set on the validation of the visual-based algorithms for close-proximity operations. These test are carried out in the Deep-space Astrodynamics Research & Technology Lab (DART Lab) Robotic Arm Facility For Autonomous cubesat ExpLoration in cLose proximity Operations (RAFFAELLO) [10] facility at Politecnico di Milano.

The different areas of interest for the Literature Review can be condensed in three main categories:

1. **Mission Review**: which are the typical trajectories of close-range proximity operations?
2. **Facility Review**: what types of facilities are used to simulate the real environment for Verification and Validation (V&V)? Which are the most important existing facilities?
3. **Robotics Fundamentals**: how is the problem modeled, from a robotic viewpoint?

1.1. Mission Review

Concerning the mission review, we have to take into account the different missions that performed close proximity science on small celestial bodies. The main identified missions can be found in the official National Aeronautics and Space Administration (NASA) Jet Propulsion Laboratory (JPL) catalogue[1]. These results have been filtered for missions that performed close proximity operations around small celestial bodies, such as asteroids, comets or dwarf planets. The selected missions include:

Mission	Type	Target	Arrival Date
NEAR	orbiter/landing	433 Eros (A898 PA)	2000-Feb-14
Stardust	sample-return	81P/Wild 2	2004-Jan-02
Hayabusa (MUSES-C)	sample-return	25143 Itokawa (1998 SF36)	2005-Sep-12
Dawn	orbiter	4 Vesta	2011-Jul-16
Rosetta	orbiter/landing	67P/Churyumov-Gerasimenko	2014-Aug-06
Dawn	orbiter	1 Ceres	2015-Mar-06
Hayabusa2	sample-return	162173 Ryugu (1999 JU3)	2018-Jun-27
OSIRIS-REx	sample-return	101955 Bennu (1999 RQ36)	2018-Dec-31
OSIRIS-APEX	orbiter	99942 Apophis (2004 MN4)	2029-Apr-13
Hayabusa2 Ext. Mission	orbiter	(1998 KY26)	2031-Jul

Table 1.1: Missions performing close proximity operations around small celestial bodies.

Focusing now on missions that performed meaningful science phases during extended bound orbital operations around their small-body targets: **NEAR**, **Dawn**, **Rosetta** and **OSIRIS-REx** have been selected for a deeper analysis.

1.1.1. Flown missions

NEAR

The Near Earth Asteroid Rendezvous (NEAR) Shoemaker mission was launched on February 17, 1996 and was the first mission to orbit around an asteroid [8].

The achieved orbits were progressively reduced from the initial capture orbit, characterized

[1]NASA, Small-Body Targets of Spacecraft Missions. Available at `https://ssd.jpl.nasa.gov/sb/targets.html`

by a periapsis distance of 321 km, down to the final close-approach pass at a distance of 2.74 km from the surface of asteroid 433 Eros (A898 PA), which has a diameter of 16.84 km, as stated in the NASA Small-Body Database Lookup [20]. Several science phases were conducted during the mission, starting with the *high orbit phase*, in which a 50 km × 50 km orbit enabled global surface mapping at a spatial resolution of 5-10 m using the NEAR camera, while simultaneously supporting scientific observations. Subsequently, the orbit was lowered to a 35 km × 35 km configuration for several days.

The trajectory was finally transitioned to an elliptical orbit in order to limit the number of close passes. The final flyovers, at altitudes of 6 km and subsequently 2.74 km, allowed the mapping of selected surface landmarks with a spatial resolution of up to 0.5 m per pixel with the NEAR MultiSpectral Imager (MSI) [2, 8, 32]. The MSI was a 537 pixel × 244 pixel charge-coupled device camera, with a Field Of View (FOV) of 2.93° × 2.25° and a pixel resolution of 96 µrad × 162 µrad [14, 30].

Dawn

The Dawn spacecraft, launched on September 27, 2007, visited asteroid 4 Vesta and dwarf planet 1 Ceres. A key role in the navigation subsystem was played by optical navigation, which was employed for orbit determination and for the retrieval of physical characteristics of the asteroids.

The camera payload comprised three main scientific instruments: the visible-light Framing Camera (FC), the Visible and Infrared Spectrometer (VIR), and the Gamma Ray and Neutron Detector (GRaND) [1]. Regarding the FC, two identical cameras were flown: FC1 and FC2. These were used during science phases as well as for Optical Navigation (OpNav). Both cameras featured a 19 mm aperture, an Instantaneous FOV (IFOV) of 93.3 µrad, and refractive optics. The incoming light was focused onto a 1024 × 1024 frame-transfer Charge Coupled Device (CCD) detector [18].

A total of four Vesta phases can be identified: *Survey*, *HAMO-1* (High-Altitude Mapping Orbits [27]), *LAMO* (Low-Altitude Mapping Orbit [27]) and *HAMO-2*.

Phase	Optimized Instrument	Orbit	Beta angle
Survey	VIR	Polar, 3000 km mean radius	12°-15°
HAMO-1	FC	Polar, 900 km mean radius	30°
LAMO	GRaND	Polar, 450 km mean radius	45°
HAMO-2	FC	Polar, 900 km mean radius	From 34° to 27°

Table 1.2: Dawn phases summarized (at Vesta) [12, 18].

Table 1.2 summarizes all the main science phases of the mission during its encounter with Vesta, focusing on the orbit type and on the Beta angle, which is the angle between the orbit plane and the Sun. It is worth noting that Main-belt Asteroid 4 Vesta has a diameter of 522.77 km [20]. In the table, can also be noted how the different phases have been optimized for specific instruments.

Focusing now on 1 Ceres, which is a Dwarf planet with a diameter of 939.4 km [20], Table 1.3 summarizes the main science phases.

Phase	Optimized Instrument	Orbit	Beta angle	FC spatial resolution
RC3	FC	circular, polar orbit 13 600 km altitude	-	1.3 km/pixel
Survey	VIR	circular, polar orbit 4425 km height	-	415 m/pixel
HAMO	FC	circular, polar orbit 1475 km height	24° to 36°	140 m/pixel
LAMO/XMO1	GRaND	355 km - 410 km height	45°	35 m/pixel

Table 1.3: Dawn phases summarized (at Ceres) [26, 33].

The complete set of mission phases, conducted at 1 Ceres, is the following: *RC3*, *survey*, *HAMO*, *LAMO/XMO1*, *XMO2*, *XMO3*, *XMO4*, *XMO5*, *XMO6* and *XMO7*. Apart from the most important shown in Table 1.3, the eXtended Mission Orbit (XMO) ranging from XMO2 to XMO7 saw progressively increasing orbit dimensions from XMO2 to XMO5 (reaching an orbit of 5400 km × 38 000 km), while it was chosen to reduce them again in XMO6 and XMO7: final orbit altitude of 4000 km × 35 km. This allowed to perform the last science measurements and gather additional images of the surface, allowing also for a natural de-orbit time grater than 20 years [26].

Rosetta

The Rosetta Spacecraft, launched March 02, 2004, targeted comet 67P/Churyumov-Gerasimenko, which has a diameter of 3.4 km [20].

The spacecraft hosts two navigation cameras (NAVCAM) with a FOV of 5° and a pixel angular size of 5 millidegree. Furthermore, the Narrow Angle Camera (NAC), with a FOV of 2.2° and a pixel size of 1.1 millidegree, and Wide Angle Camera (WAC), with a FOV of more than 10° and a pixel size of 5.8 millidegree, of the OSIRIS scientific instrument are also used for navigation purposes [11].

Regarding the mission, Rosetta performed a total of 5 mission phases around the comet: Close Approach Trajectory (CAT), Transition to Global Mapping Phase (TGM), Global Mapping Phase (GMP), Close Observation Phase (COP) and the Science surface package Delivery Phase (SDP). Their characteristics are reported in the following table:

Phase	Trajectory type	Size
CAT	hyperbolic arcs	distance from 100 km to 60 km
TGM	-	-
GMP	bound orbits	radius of 30 km
COP	terminator orbits	radius from 20 km to 10 km
SDP	bound orbit	radius of 30 km

Table 1.4: Rosetta mission phases [11].

OSIRIS-REx

The last analyzed mission, NASA Origins, Spectral Interpretation, Resource Identification, and Security-Regolith Explorer (OSIRIS-REx), was launched September the 8, 2016 and performed close range proximity operations around Apollo asteroid 101955 Bennu (1999 RQ36). 101955 Bennu has a size of 0.484 44 km of diameter [20].

The spacecraft was equipped with several scientific payloads, but the most important, regarding visual systems, are the OSIRIS-REx Camera Suite (OCAMS) and the Touch And Go Camera System (TAGCAMS). OCAMS comprised three cameras: MapCam, PolyCam and SamCam, which together fulfilled all imaging requirements from 500 km down to 2 m above the surface.

Camera	Specifications	Type
MapCam	focal length of 125 mm, FOV 4°	medium-field
PolyCam	focal length of 630 mm	narrow-field
SamCam	focal length of 24 mm	wide-angle

Table 1.5: OSIRIS-REx OCAMS [16].

The three cameras reported in Table 1.5 were used in different phases, according to their capabilities: MapCam was used during the approach phase, to search for natural satellites and dust plumes, images needed for base maps, global shape model and spin-state measurements. PolyCam and SamCam, were employed, in order, for close scrutiny of the sample sites and for recording the sampling event. The other payload, TAGCAMS, was instead used to aid navigation and photo documentation around Bennu. The system comprised three cameras: NavCam 1, NavCam 2 and StowCam, all of which had a 2592 pixel × 1944 pixel detector array, with a pixel depth of 12 bit. Furthermore, all these cameras shared a FOV of 44° × 32°, with a pixel scale of 0.28 mrad/pixel [16].

Regarding the mission review, OSIRIS-REx performed different types of trajectories: *Baseball Diamonds* (hyperbolic flybys) and elliptical orbits. The Proximity Operations at Bennu can be reconstructed in this order [3]:

Campaign	Activities and Orbital Characteristics
A. Navigation Campaign	• **Approach** • **Preliminary survey:** 7 km hyperbolic flybys over northern, equatorial and southern regions • **Orbital-A:** 2.1 km × 1.5 km elliptical frozen orbit
B. Site Selection Campaign	• **Detailed survey:** hyperbolic flybys at distances from 3 km to 5 km at different longitudes • **Orbital-B:** 925 m near-circular polar orbit • **Orbital-C:** frozen Sun-terminator 1.9 km × 1.6 km orbit • **Detailed Survey Flyby 2, Re-flight and Reconnaissance-A:** low-altitude flyovers between 1 km and 1.25 km, generation of a 2 cm/pixel Digital Terrain Map (DTM) • **Orbital-R:** frozen 1.1 km × 1.4 km orbit • **Site selection**

Continued on next page

Campaign	Activities and Orbital Characteristics
C. Sample Acquisition Campaign	• **Reconnaissance B & C** • **Checkpoint Rehearsal:** 0.9 km × 1.2 km counter-clockwise orbit • **Episode XIII, Reboot:** 0.8 km × 1.0 km clockwise orbit • **Matchpoint rehearsal** • **Touch And Go (TAG)**
D. Post-TAG Operations	• **Post-TAG Observations:** 3 km flyby • **Asteroid Departure Maneuver**

Table 1.6: OSIRIS-REx Campaigns and activities [3].

The last two campaigns of the mission (*E. Bennu Ephemeris, Geophysics & Shape Model Development*; *F. Gravity*) are not reported in Table 1.6, since no orbital maneuvers were performed.

Table 1.7 shows a summarized version of the relevant data gathered in the previous analyzed missions.

Mission	Trajectories	Minimum distance[2]	Diameter of target
NEAR	Orbits	2.74 km height	16.84 km
Dawn: 4 Vesta	Orbits	450 km radius	522.77 km
Dawn: 1 Ceres	Orbits	35 km height	939.4 km
Rosetta	Orbits and hyperbolic arcs	10 km radius	3.4 km
OSIRIS-REx	Orbits and Baseball Diamond	0.8 km radius	0.484 km

Table 1.7: Trajectory types and data of selected missions.

[2]Either referring to the radius of the orbit or to the height above the target

1.2. Facility Review

An overview of the facilities is now presented. This is carried out to investigate the hardware used by other research groups, and to compare the RAFFAELLO facility.

1.2.1. Testbed types

The existing testbeds, which are being used to test GNC algorithms, can be divided in several families, depending on how physical elements and software are used to simulate a mission.

A study from Stanford University [4] subdivided HIL testbeds into different types, regarding to what extend the facility is able to reproduce real data:

1. **Digital Twin** Often developed via proprietary tools, commercial software is usually limited due to its highly specialization on specific applications. Digital Twinning methodologies allow to perform missions planning with modern software including GMAT[3] and FreeFlyer[4]. Other software like Basilisk, MATLAB and Simulink[5] can be employed for GNC V&V.

2. **Robotic Twin** Software simulations may be inaccurate for physical testing and, for this reason, HIL tests are needed for validation. The main categories are:

   - **GNSS Simulators:** Typically used to test sensors for navigation, these represent the higher level of abstraction since environment data is generated via software.

   - **Optical Simulators:** Used to test optical sensors such as star trackers, often employed to test visual based Guidance algorithm: the whole trajectory and environment are generated via software, while the simulated camera FOV is rendered and projected on a screen, where the testbed camera can capture the close-to-reality image.

   - **Kinematic Testbeds:** High-torque actuators are employed to impose the simulated or real motion of one or more bodies. Often used for close proximity operations, such as docking or in-orbit servicing, can be used in a scaled environment to simulated different missions [31].

[3] NASA, General Mission Analysis Tool (GMAT). Available at `https://etd.gsfc.nasa.gov/capabilities/capabilities-listing/general-mission-analysis-tool-gmat/`

[4] a.i. solutions, FreeFlyer. Available at `https://ai-solutions.com/`

[5] MathWorks, Simulink. Available at `https://www.mathworks.com/products/simulink.html`

- **Dynamics Testbeds:** Entire mission is being tested, using actuators to reproduce the dynamics and test their performances. This can allow to completely validate GNC subsystems [31].

HIL is extremely important in scenarios where real data is lacking and flight test are extremely expensive or simply inaccessible.

1.2.2. Existing Kinematic Testbeds

The existing facilities are designed upon specific requirements: starting from small testbeds where only sensors are being tested, arriving to full-scale mockups of spacecraft controlled via robotic manipulators. RAFFAELLO is a Kinematic Testbeds.

The focus of this thesis is the analysis and representation of spacecraft trajectories around small bodies such as Didymos and Dimorphos; for this reason, only specific testbeds are considered in this section. These are facilities in which proximity operations are tested, such as Rendezvous and Capture (RvC), rendezvous and docking (RvD) or similar missions. For this reason, only Kinematic Testbeds are analyzed.

Table 1.8 shows all the major Kinematic Testbeds used for proximity operations. While all of them are used for HIL purposes, the Space Operations Simulation Center (SOSC) facility is not of interest, since the scale and the hardware used are extremely different from all the other facilities.

Facility	Institution	Location
EPOS 2.0	DLR	Germany
SOSC	Lockheed Martin	USA
TRON	DLR	Germany
TRON (SLAB)	Stanford University	USA
INVERITAS	DFKI RIC	Germany
Zero-G Lab	University of Luxembourg	Luxembourg

Table 1.8: Major Kinematic Testbeds regarding proximity operations.

EPOS 2.0

Starting with the European Proximity Operations Simulator (EPOS) 2.0 [5], it is located at the German Space Operation Center (GSOC) in Oberpfaffenhofen (Germany) and it is used for RvD purposes. Its name recalls the original EPOS 1.0, which was a joint test facility created by Deutsche Forschungsanstalt für Luft- und Raumfahrt (DLR) and

European Space Agency (ESA). The new facility, instead, was created between 2008 and 2009 and started from the work of two institutes of DLR: GSOC and Robotics and Mechatronics Institute of DLR.

The facility is composed by two 6-DOF KUKA[6] robotic manipulators and a linear slide of 25 m, where one of the robot is mounted and gains one additional DOF, given by the translation. The lighting environment is simulated via an ARRI[7] Max 12/18, which is capable of generating a luminous flux of 1.15×10^6 lm.

The structure of the facility is well suited for docking simulations, since the linear DOF is able to move the relative distance of the robots from 25 m to 0 m. Regarding the hardware specifications, the following tables condense the data sheets:

Type	Payload mass (maximum)	Rail mass (without robot)	Repeatability
KUKA KL1500 (rail)			
Linear axis with rack-and-pinion drive	3000 kg	ca. 13 500 kg	±0.02 mm (ISO 9283)
KUKA KR100HA (robot 1), rail mounted			
6-axis articulated robot	100 kg	1200 kg	±0.12 mm (ISO 9283)
KUKA KR240-2 (robot 2), floor mounted			
6-axis articulated robot	240 kg	1267 kg	±0.12 mm (ISO 9283)

Table 1.9: EPOS 2.0 hardware configuration [5].

Regarding the light, an ARRI Max 12/18 is equipped with a 12 kW Hydrargyrum Medium-arc Iodide (HMI) Osram[8] lamp, capable of outputting a luminous flux of 1.15×10^6 lm.

A large protective fence defines the outer safe zone, which must be locked during simulations, for safety reasons. If this criteria is not met, the facility can be only controlled via KUKA control panels (KCPs).

These type of facilities, which cannot emulate the dynamic part of the simulation, rely on software simulations for orbital dynamics and trajectory simulations.

[6]KUKA, KUKA Industrial. Available at `https://www.kuka.com/en-us/products/robotics-system/industrial-robots`

[7]ARRI, ARRI lighting. Available at `https://www.arri.com/en/lighting`

[8]OSRAM, OSRAM Lamps. Available at `https://www.osram.com/cb/`

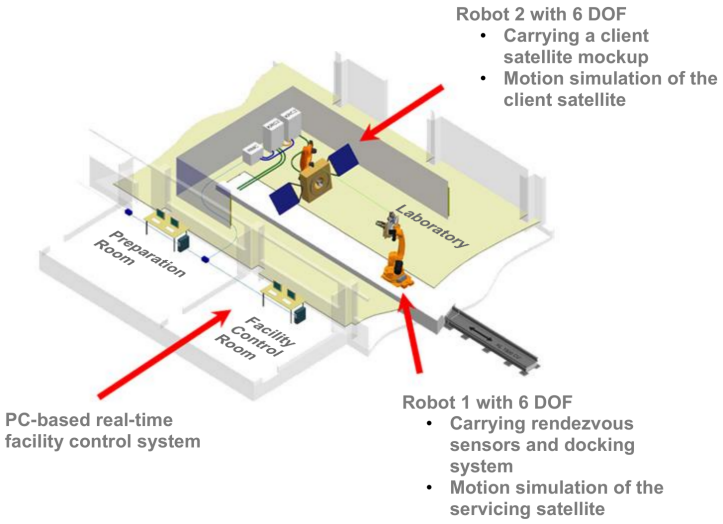


Figure 1.1: EPOS 2.0 testbed - *Taken from [5] under CC BY 4.0 License.*

TRON

DLR's Testbed for Robotic Optical Navigation (TRON) [15, 34] supports the development and validation of optical navigation systems, enabling a validation of camera based sensors up to a TRL of 7.

Facility hosts a KUKA KR16-2 robotic manipulator, installed on a 11 m rail in the 15 m × 5.10 m × 3.00 m dedicated room. Its maximum end-effector payload of 16 kg can be extended to 40 kg with an additional robot base mass. The static repeatability of the robot is ±0.1 mm. Regarding the target body, TRON walls are covered with multiple scaled Moon terrain mockups. Facility also provides a rotating 3D scale of asteroid 433 Eros.

Moving curtains are used to exclude any unwanted light source from interfering with the facility and black paint is used on all the remaining surfaces. Lighting is extremely important and the testbed uses a zoom profile spotlight ADB WARP, with a HMI technique to achieve a 6000 K color temperature. This lamp, which is capable of rotating along two axis, is mounted on a 3-DOF gantry, which complete the total 5-DOF of the lighting system.

One unique element of TRON is the ability to perform test both in open and close loop, via sensor feedback, in real-time.

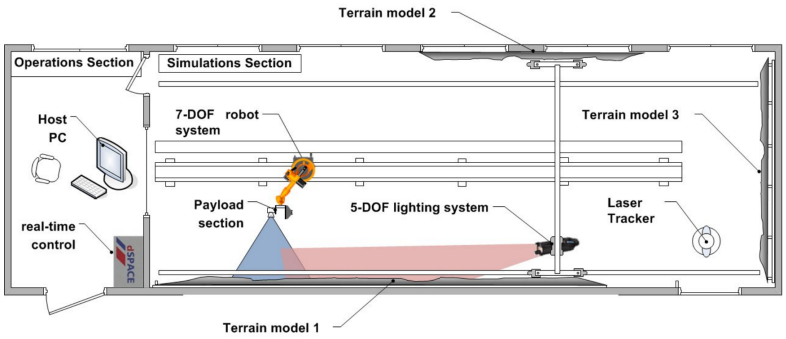


Figure 1.2: TRON (DLR) testbed - *Taken from [15] under CC BY 4.0 License.*

TRON (SLAB)

The Space Rendezvous Laboratory (SLAB) at Stanford University hosts the Testbed for Rendezvous and Optical Navigation (TRON) [22] facility, where vision-based rendezvous trajectories are simulated in a 8 m × 3 m × 3 m simulation room.

The testbed consists of two 6-DOF KUKA robotic arms (KR 10 R1100 sixx C-WP): one supports the camera and it is mounted on the ceiling, while the other holds the target Resident Space Object (RSO) and is mounted on the floor. The first one, is also installed on a rail: providing an additional DOF and allowing a maximum distance of 6 m between the two robots.

The RSO target can be manufactured with two mounting spots, at opposite sides, in order to be mounted relative to the desired configuration, without robot part obstructing the camera.

12 Vicon Vero[9] cameras are mounted in the facility, tracking the Infrared (IR) markers of the end-effectors of the robots and giving its true position in space. Additionally, the KUKA robots outputs the telemetry of the poses of both arms' end-effectors based on their joint angles. This is crucial to calibrate the facility.

Regarding the illumination, TRON uses two different systems: an array of 10 light boxes and a halide arc lamp. The latter is used to simulate direct sunlight, while the others are used to emulate Earth albedo, using hundreds of LED strips covered with a diffuser plate for each light box. Light-absorption black commando curtains are used to block windows and non active light boxes.

[9]Vicon, Vicon Vero. Available at `https://www.vicon.com/hardware/cameras/vero/`

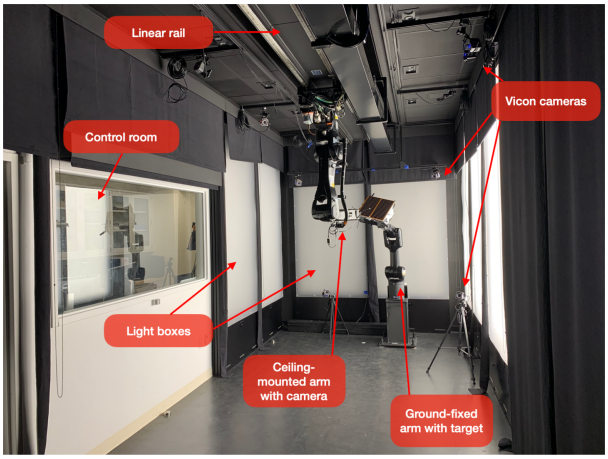


Figure 1.3: TRON (SLAB) testbed - *Taken from [22] under CC BY 4.0 License.*

INVERITAS

The Inveritas [23, 24] facility is located in Kaiserslautern (Germany), in the Space Exploration Hall of the Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI) Robotics Innovation Center (RIC). The testbed is able to replicate absolute poses of client and servicer satellite mockups, generated by a real-time software simulation.

The architecture for this test facility is dictated by space allocation problems: the testbed serves as RvC simulations, but a solution similar to EPOS 2.0 couldn't be build given the presence of an artificial lunar crater mockup in the facility. For this reason, Inveritas comprises a 6-DOF robotic arm (for the Client satellite) and a 4-DOF cable robot (for the Servicer satellite). This allows for a large range of motion without interfering with existing hardware. Furthermore, this combination is also capable of simulating landing maneuvers on the lunar surface.

The total number of DOF is given by the robotic manipulator, which has 6 DOF, and by the cable robot, which has 3 translation DOF and 1 rotational (along the vertical axis): for a total of 10 DOF. It can be noted how the DOF of 2 satellites moving in space is 12, which is greater than the available DOF of the facility: this, however, is not a problem, since the desired map of the facility takes into account the relative motion of the two bodies, and not their individual trajectories.

Regarding the hardware specifications, the robotic manipulator is a KUKA KR60-3, with a maximal payload of 60 kg (at a lever of 200 mm). This arm is controlled via Ethernet at a frequency of 83.33 Hz and it can be controlled both in Cartesian and joint space. The cable robot can move freely in the 3D space and thus use a diagonal approach of up to 16.5 m inside the available space, which is hosted in the 24 m long, 12 m wide and 10 m

high facility. This custom cable robot has been developed by Spidercam and can host a payload of up to 150 kg, while being controlled through a 10 Gbit s^{-1} optical wire at 250 Hz.

The lighting system is able to replicate almost parallel (collimated) light beams which are needed to replicate hard shadows and strong differences between illuminated and dark surfaces. The facility is equipped with 6 mobile spotlights, each one is motorized, allowing pan and tilt rotations and varying the FOV from 12° to 30°. Their position can also be adjusted from 1 m to 6 m from ground. A 6000 K light is delivered via the 575 W gas discharge lamps, with a maximum intensity of 14 500 lx at 10 m of distance and a FOV of 12°. Moreover, a special light absorption paint is used to darken walls, ceiling and visible part of the facility.

A Motion Tracking System (MTS) is used to perform offline position error evaluations, thanks to the 7 infrared cameras made by Vicon and a non-symmetric setup of passive IR markers.

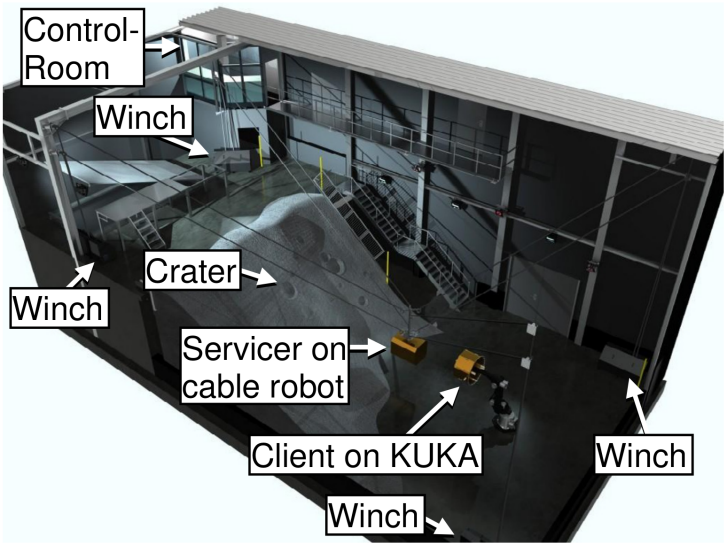


Figure 1.4: Inveritas testbed - *Taken from [23] under CC BY 4.0 License.*

Zero-G Lab

The Zero-G Lab [21] is located at the Interdisciplinary Centre for Security, Reliability and Trust (SnT), in the University of Luxembourg, Kirchberg, Luxembourg. The facility is composed by two 6-DOF robotic manipulators and two floating platforms. The two floating platforms can move on a frictionless floor thanks to their air cushion, while, the two robots are mounted on the ceiling and on the wall of the facility, along with their respective rails. The facility has a volume of 7 m × 6 m × 2.30 m, while the inner safe

testing zone is 5 m × 3 m × 2.30 m (WxLxH) and it is painted in non reflective black.

The two robotic manipulators (Universal Robots UR10e[10]) allow for a total of 14 DOF, proving their ability to map two distinct objects in the complete spatial movements, including the possibility to add a significative relative distance. With a working radius of 1300 mm and the controllability given by the Robot Operating System (ROS) Network, they are integrated on two rails capable of translating of 3204 mm and 4330 mm for the ceiling and wall mounted rails respectively.

The verification and validation phase is carried out using an under-millimeter-precision motion capture system running at 240 Hz.

Regarding the illumination problem, Zero-G Lab uses a Godox SL-60[11] Video Light 60 W LED lamp with a temperature of 5600 K (corresponding to daytime sunlight). The sizing aimed at reproducing the same irradiance of Sun on Earth's orbit (1.35 kW m^{-2}). The final system comprises two light source modifiers, a collimator and a reflector for different scenarios and the light can both be mounted on a robotic manipulator or placed static in the facility.

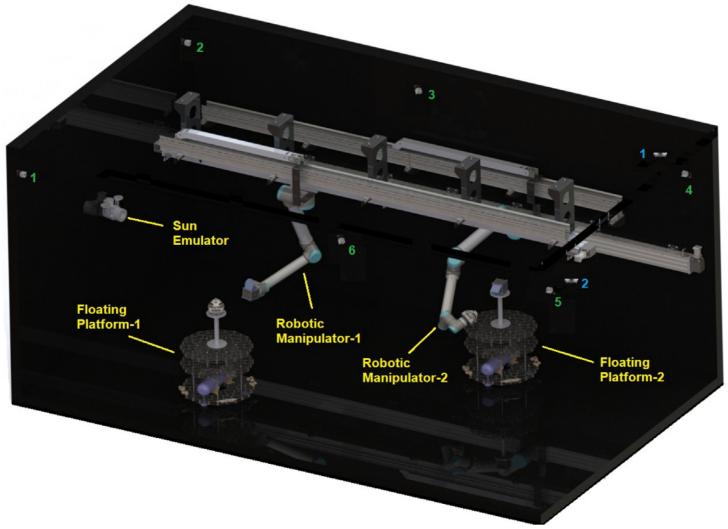


Figure 1.5: Zero-G Lab testbed - *Taken from [21] under CC BY 4.0 License.*

Facility Review Overview

From the facility review, it is clear how the facilities are all built similar: high torque robotic manipulators are used to impose the dynamics, while high fidelity mockups of

[10]Universal Robots, Universal Robots UR10e. Available at `https://rbtx.com/en-US/components/robots/universal-robots-ur12e-6dof-1300mm-125kg`

[11]Godox, Godox SL60W. Available at `https://www.godox.com/product-d/SL60.html`

target and servicer spacecraft reproduce the hardware to be validated.

Regarding lighting conditions, a professional setup allow to isolate the test facility from external light sources and specifically designed lamps are employed to emulate sunlight depending on the scenario.

Finally, a motion tracking system allows to retrieve the positions of the mockups with high precision, to better calibrate the facility or to evaluate performances.

One important note concerns the software interface: no publicly available data describe how the software-generated trajectories are mapped within the facility, apart from a few hints; therefore, a thorough development of such algorithms will be carried out.

1.2.3. RAFFAELLO testbed

The RAFFAELLO facility, located in the DART Lab at Politecnico di Milano, is composed of two 6-DOF robotic manipulator, which can be used to test the vision based navigation algorithms via HIL testing.

It consists in a double Yaskawa MOTOMAN GP12[12] manipulators, one of which is mounted on a linear rail: however, the current RAFFAELLO configuration do not use this additional degree of freedom, and its implementation has been discarded in the thesis. The facility is bounded by a blacked-out cage, and a lamp, mounted on a tripod, is present to emulate Sun light.

Regarding the lamp, a Godox SL60W is used for this scope: with an illuminance of 4100 lx (at 1 m and 100 % power) and a color temperature of 5600 ± 300 K.

In order to acquire photos of the target, RAFFAELLO hosts a Basler scA1300-32gm[13]: a camera with a sensor size of 1296 × 966 pixel.

1.3. Robotics Fundamentals

The RAFFAELLO facility hosts two Yaskawa MOTOMAN GP12 robotic manipulators. These robots could be modeled via MATLAB Robotics System Toolbox[14], but a custom

[12]YASKAWA, GP12 Product Page. Available at `https://www.yaskawa.eu.com/robotics/robots/handling-mounting/productdetail/product/gp12_693`

[13]Basler, Basler sca1300-32gm datasheet. Available at `https://assets-ctf.baslerweb.com/dg51pdwahxgw/7uBPu2c1Uaf4pfsLl6v8Be/a11f70b33cc83b9a70ada92f25ec8c78/AW00011919000_scout_GigE_User_Manual.pdf`

[14]MathWorks, Robotics System Toolbox. Available at `https://mathworks.com/products/robotics.html`

model was chosen, in order to obtain more control on the hardware and more flexibility in the case of an optimization, where standard MATLAB functions may not be suitable.

From the GP12 datasheet [35], a complete schematics of the robots can be used to develop a digital twin of the system and to solve Forward Kinematics (FK) and Inverse Kinematics (IK).

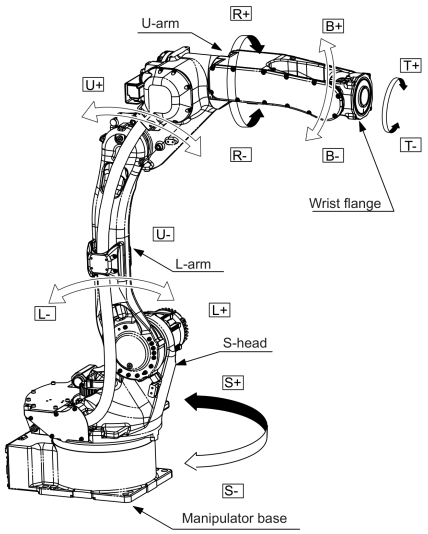


Figure 1.6: GP12 robotic manipulator - *Taken from GP12 instructions [35] under CC BY 4.0 License.*

Figure 1.6 shows the structure of the robot, highlighting all the rotational joints, including their positive rotation direction. Starting from the base and going towards the end effector, the joints are: S, L, U, R, B and T.

It can be immediately noticed how the robot can be categorized as a *6R* manipulator, since all 6 joints are *revolute (R)* joints and none are *prismatic (P)*. Furthermore, the roles of the joints are given in this order: Base rotation (S), Shoulder (L), Elbow (U), Wrist roll (R), Wrist pitch (B) and Wrist yaw (T). This structure has 6-DOF and can place and orient an end-effector arbitrarily in 3D space, simplifying also the IK thanks to the last three joints which form a spherical wrist.

Lastly, a CAD drawing of the robot can be downloaded from the GP12 product page. This will be useful to check for symmetries and sizes of each link (a rigid body that connects two consecutive joints in a manipulator and transmits motion and forces between them).

Table 1.10 summarizes all the physical limitation of the GP12, as found in the official instructions [35].

GP12	Axis	Specification
Joints range	S-Axis	$[-170°, 170°]$
	L-Axis	$[-90°, 155°]$
	U-Axis	$[-85°, 150°]$
	R-Axis	$[-200°, 200°]$
	B-Axis	$[-150°, 150°]$
	T-Axis	$[-455°, 455°]$
Maximum Speed	S-Axis	$4.53\,\mathrm{rad\,s^{-1}}$ $(260\,°\,\mathrm{s^{-1}})$
	L-Axis	$4.01\,\mathrm{rad\,s^{-1}}$ $(230\,°\,\mathrm{s^{-1}})$
	U-Axis	$4.53\,\mathrm{rad\,s^{-1}}$ $(260\,°\,\mathrm{s^{-1}})$
	R-Axis	$8.20\,\mathrm{rad\,s^{-1}}$ $(470\,°\,\mathrm{s^{-1}})$
	B-Axis	$8.20\,\mathrm{rad\,s^{-1}}$ $(470\,°\,\mathrm{s^{-1}})$
	T-Axis	$12.2\,\mathrm{rad\,s^{-1}}$ $(700\,°\,\mathrm{s^{-1}})$
Allowable Moment	R-Axis	$22\,\mathrm{N\,m}$
	B-Axis	$22\,\mathrm{N\,m}$
	T-Axis	$9.8\,\mathrm{N\,m}$
Allowable Inertia ($\mathrm{GD^2/4}$)	R-Axis	$0.65\,\mathrm{kg\,m^2}$
	B-Axis	$0.65\,\mathrm{kg\,m^2}$
	T-Axis	$0.17\,\mathrm{kg\,m^2}$

Table 1.10: GP12 - Basic Specifications [35].

1.3.1. Forward Kinematics

The problem of determining the position and attitude (i.e., the pose) of the end effector, given the joints solution $\boldsymbol{\theta}$, is called FK [17]. This problem is extremely trivial and consists in calculating consecutive translations and rotations of joint positions and frames. This can be achieved in two main ways: by using the Denavit-Hartenberg (D-H) parameters [7] (homogeneous transformation matrices) or with the product of exponentials (PoE) formula [17].

It is important to define the difference between **Cartesian** and **joint space**: the latter refers to the configuration space defined by the robot's joint coordinates (e.g., angles of each joint), while Cartesian space represents the pose of the end-effector in the task (or physical) space.

The two main ways of representing the FK are the PoE and the D-H parameters. The D-H convention assigns a reference frame to each link and uses only four parameters per

joint to describe the relative pose from the previous frame, resulting in a compact and easily implementable chain of homogeneous transformations: ideal for numerical coding and debugging. The PoE approach, instead, formulates the kinematics as a product of exponential of screw axes (twists), defining only the base frame, the end effector frame and one screw axis per joint. Although the PoE formula appears to be the chosen representation for Kevin Lynch and Frank Chongwoo Park [17], the D-H parameters allow for faster and simpler implementation and debugging. For this reason, the D-H notation has been chosen for the FK.

In the D-H notation, each joint is defined by a rotation matrix and a joint position. The first is given by the rotation axis of the joint, while the latter is defined by the absolute position of the joint with respect to the previous one. The structure of these homogeneous matrices is:

$$\boldsymbol{T}_{i\to i+1} = \begin{bmatrix} \boldsymbol{R}_{i\to i+1} & \boldsymbol{l}_{i\to i+1} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \in \mathrm{SE}(3), \quad i = 0, \dots, n-1 \tag{1.1}$$

where:

- n is the total number of joints of the robot.
- $\boldsymbol{R}_{i\to i+1} \in \mathrm{SO}(3)$ is the 3×3 rotation matrix mapping vectors from frame i to frame $i+1$, i.e.,

  $$\boldsymbol{v}_{i+1} = \boldsymbol{R}_{i\to i+1}\, \boldsymbol{v}_i$$

  for any non-homogeneous vector $\boldsymbol{v}_i$ expressed in frame i.
- $\boldsymbol{l}_{i\to i+1} \in \mathbb{R}^3$ is the position vector of the origin of frame $i+1$ with respect to the origin of frame i, expressed in the coordinates of frame i.

The homogeneous transformation $\boldsymbol{T}_{i\to i+1}$ encodes both rotation and translation in a single matrix. For any point $\boldsymbol{x}$ expressed in frame i, the corresponding point in frame $i+1$ is

$${}^{i+1}\boldsymbol{x}^h = \boldsymbol{T}_{i\to i+1} \begin{bmatrix} {}^{i}\boldsymbol{x} \\ 1 \end{bmatrix} \in \mathbb{R}^4$$

where the superscript ^{h} indicates the *homogeneous* notation, given ${}^{i+1}\boldsymbol{x}^h = \begin{bmatrix} {}^{i+1}\boldsymbol{x} \\ 1 \end{bmatrix}$.

For a serial manipulator with n joints, the pose of the end-effector (frame n) with respect to

the base frame (frame 0) is obtained by the **product of consecutive transformations**:

$$\boldsymbol{T}_{0\to n} = \prod_{i=0}^{n-1} \boldsymbol{T}_{i\to i+1} \tag{1.2}$$

This composition of transformations provides the FK of the manipulator. In particular, for any point ${}^0\boldsymbol{x}$ expressed in the base frame, its coordinates in the end-effector frame are

$${}^n\boldsymbol{x}^h = \boldsymbol{T}_{0\to n}\, {}^0\boldsymbol{x}^h$$

Each $\boldsymbol{T}_{i\to i+1}$ contributes both rotation and translation, so their consecutive product propagates the pose from the base to the end-effector. In this way, the end-effector's global position and orientation are directly determined by the kinematic chain.

1.3.2. Inverse Kinematics

For an n degree-of-freedom open chain, the inverse kinematics problem consists in determining the joint coordinates $\boldsymbol{\theta} \in \mathbb{R}^n$ such that the forward kinematics map yields the desired end-effector pose: $\boldsymbol{T}(\boldsymbol{\theta}) = \boldsymbol{X}$ for a given $\boldsymbol{X} \in \mathrm{SE}(3)$ [17].

Solving the IK of a generic N-DOF robotic manipulator is not always straightforward and often does not admit closed-form solutions. This is not the case for the GP12: since the last three joints, R, B, and T, form a *spherical wrist* [17], meaning that the rotation axes of these three joints intersect at a single point.

The *spherical wrist* allows the definition of a point in space that remains fixed for an arbitrarily chosen configuration of joints R, B, and T (i.e., the intersection point of the last three joint axes).

For this reason, the problem of finding the right combination of $\boldsymbol{\theta}$ which generates the desired $\boldsymbol{T}(\boldsymbol{\theta})$, can be subdivided in two different problems, which can be decoupled and solved one at the time: **position problem** and **orientation problem**.

1. **Position problem:** This problem focuses on finding the solution of the first three joints, in order to reach the desired location of the *spherical wrist*, which is independent from the last three joints solution.
2. **Orientation problem:** This problem focuses on finding the last three joints, to correctly represent the desired attitude, assuming the first three joints have already been determined.

1.4. Research Question

It is clear how each facility creates their own robotic testbed, based on specific requirements and depending on the type of data to be simulated. For this reason, there is no general mapping of the real trajectories and each research group creates a custom suited algorithm.

The main question arises: *To what extent is it possible to use a robotic manipulator testbed to generate accurate, dynamically consistent trajectory data that mimic real-life motion profiles?*

In this context, dynamic consistency means preserving the key kinematic and geometric characteristics of the original orbital problem: namely correct relative positions and velocities, accurate attitudes and realistic Sun illumination direction. The simulated data should remain meaningful for relative navigation.

This research question is tied to the RAFFAELLO configuration, but its implementation is prepared to all the possible future modifications. Furthermore, the scope of the thesis is to investigate and produce a mapping methodology, which can be used for any given trajectory, target and phase angle configuration.

2 | Methodology

The Development chapter of the thesis defines all the implementations and technical details regarding the robotic advances, the facility layout and minor changes as well as the main scope of this document: the facility optimization.

Thorough these sections, the hardware and software implementations will be analyzed: for this reason, a notation paragraph will be present whenever it is crucial for the documentation.

A brief explanation on the notation is presented:

- vectors are defined by lowercase, bold letters, such as $\boldsymbol{r}$. Superscript notations like ${}^{N}\boldsymbol{r}$ indicates the reference frame where the vector is defined: inertial, in this example;
- unit vectors are defined by lowercase, bold letters with hat, such as $\hat{\boldsymbol{r}}$. Superscript notations like ${}^{N}\hat{\boldsymbol{r}}$ indicates the reference frame where the vector is defined;
- matrices are defined by uppercase, bold letters, such as $\boldsymbol{R}$;
- reference frames are defined with $\{N\}$;
- rotation matrices are defined by uppercase, bold letters with subscript like ${}_{S/N}$, such as $\boldsymbol{R}_{S/N}$: in this case, from frame $\{N\}$ to frame $\{S\}$;
- specific unit vectors, derived from the orthonormal basis of each reference frame, are denoted by $\hat{\boldsymbol{e}}_i$, with i = 1, 2, 3 corresponding to the x, y and z directions respectively;
- Joints of the GP12 are denoted using both the standard industrial names S, L, U, R, B, T and the sequential indices J1-J6, respectively.
- position of a specific point is denoted with $\boldsymbol{p}_{J1}$, where the letter $\boldsymbol{p}$ indicates the position vector and the subscript indicates the point: joint 1, in this example;
- zero matrix is denoted as $\boldsymbol{0}$, while the identity matrix is written as $\boldsymbol{I}$, with dimensions specified with subscript, when necessary.

2.1. Robotics

Starting from the Robotic implementation, FK and IK are the main focus of this section, since, for this implementation, the GP12 robotic manipulators are controlled in joint space, while trajectories and facility constraints are defined in task space.

One preliminary and important note regards the various solutions of IK: for this map, the relation between joint space and task space is not unique. In fact, for a 6R PUMA-type robotic manipulator, four different orientations of the first three joints produce the same end effector pose [17].

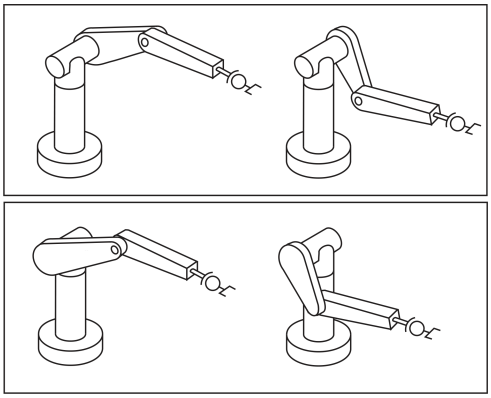


Figure 2.1: Four possible IK solutions for a 6R PUMA-type arm - *Taken from [17] under CC BY 4.0 License.*

Figure 2.1 shows this behavior, with the so called *lefty solutions* (elbow-up and elbow-down) and with the *righty solutions* (elbow-up and elbow-down): top left, top right, bottom left and bottom right pictures respectively. However, for the GP12, the only feasible solution is the righty elbow-up: the other three solutions, in fact, are non reachable, given the robot specifications shown in Table 1.10. The real problem of this non-singular solutions arises in the last three joints, where different combinations of θ_4, θ_5 and θ_6 may lead to the same end effector attitude. This behavior will be addressed in the section 2.1.3.

2.1.1. Frames and conventions

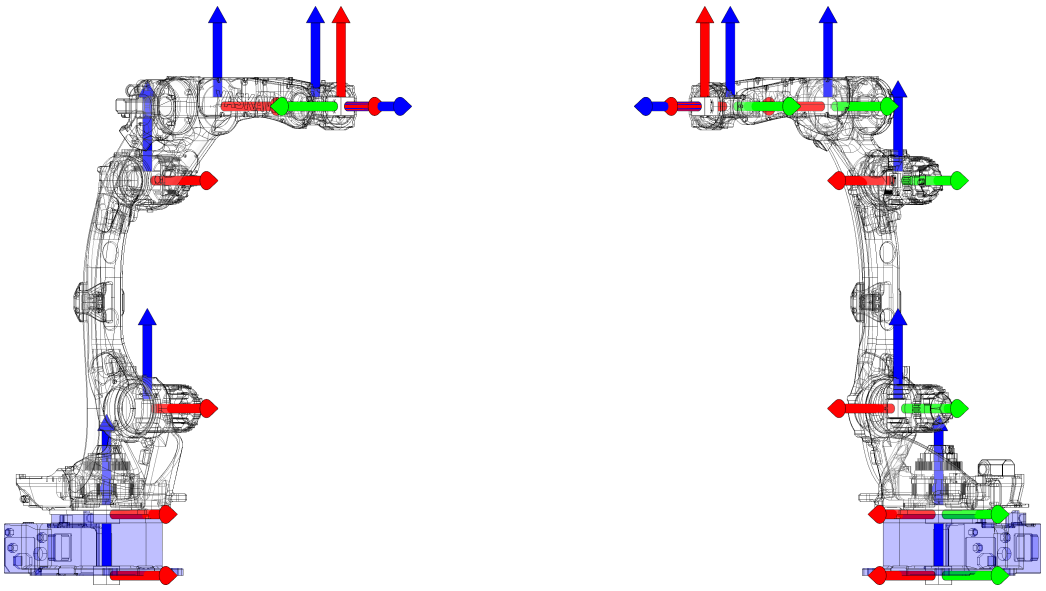
Starting from the frame definitions for this section:

- **$\{G\}$ - Ground frame** Facility reference frame, defines all the quantities related to positions and orientations of the physical elements of the testbed;
- **$\{B\}$ - Base frame** This frame is the body frame of the GP12, it is used to align the base of the robot with the facility. This frame is used to derive robotics formulations.

Each GP12 manipulator is defined by the following quantities:

- **Base pose**: $\boldsymbol{T}_{G/B}$, used to define orientation ($\{B\}$) and position of the base of the robot in the facility, it comprises a rotation matrix, which defines the orientation of the fixed base of the robot, and a position vector, which defines the location of the base. Both are defined relative to the global reference frame of the facility, called ground reference frame $\{G\}$;
- **Joint pose**: used to define orientation and position of each joint of the robot with respect to the previous one, they are defined in the *home* position of GP12, where each joint angle is equal to zero.

Unless otherwise stated, all Computer-Aided Design (CAD) models of the YASKAWA GP12 robot presented in this thesis were reconstructed by the author in SOLIDWORKS[1] using the manufacturer-provided .STEP file available on the YASKAWA GP12 product page[2].



Front-right view Front-left view

Figure 2.2: GP12 joint frames (home position).

Frames are color coded as follows: red for the x-axis ($\hat{\boldsymbol{e}}_1$), green for the y-axis ($\hat{\boldsymbol{e}}_2$) and blue for the z-axis ($\hat{\boldsymbol{e}}_3$).

[1]Dassault Systèmes - SolidWorks Corporation, SOLIDWORKS. Available at `https://www.solidworks.com/`

[2]YASKAWA, GP12 Product Page. Available at `https://www.yaskawa.eu.com/robotics/robots/handling-mounting/productdetail/product/gp12_693`

In Figure 2.2, it is possible to see both types of frames: the one underneath the base is the $\{B\}$ frame, while the other 6 frames are located in the joints of the robot. In the *home* position (when all the joint angles are equal to zero), the frames of the first 5 joints are aligned with the base frame, for convention: in this condition, $\boldsymbol{R}_{J_{i-1}/J_i} = \boldsymbol{I}_{3\times3}$, with i ranging from 1 to 5 (J0 being the base). The last frame, instead, is:

$$\boldsymbol{R}_{J5/J6} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

The rotation matrices are defined as always, by the positive right-hand rule:

$$\boldsymbol{R}_x(\beta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) \\ 0 & \sin(\beta) & \cos(\beta) \end{bmatrix} \quad \boldsymbol{R}_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix}$$

$$\boldsymbol{R}_z(\beta) = \begin{bmatrix} \cos(\beta) & -\sin(\beta) & 0 \\ \sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

One problem regards the definition of joint's positive rotations: Figure 1.6 shows how S and L are positive in the defined joints frames, while U, R, B and T are negative. However, this is not a problem, since the FK and IK are custom made, allowing to define the same joint solution for the custom developed algorithms and for the interface with the real robot.

2.1.2. Forward Kinematic

FK is based on the D-H notation defined with Equation 1.2. In order to define a function that maps the joint angles ($\boldsymbol{\theta}$) to the end-effector pose, it is necessary to specify the rotation axes and the relative positions of the reference frames.

Joint	Specification	Type	Axis	Relative position $\boldsymbol{l}$
S (J1)	turning	Revolute	z	[0 0 150] mm
L (J2)	lower arm	Revolute	y	[155 0 300] mm
U (J3)	upper arm	Revolute	-y	[0 0 614] mm
R (J4)	wrist roll	Revolute	-x	[140 0 200] mm
B (J5)	wrist pitch/yaw	Revolute	-y	[500 0 0] mm
T (J6)	wrist twist	Revolute	-x	[100 0 0] mm

Table 2.1: GP12 - joints data [35].

Using the information reported in Table 2.1, it is possible to define the complete FK by multiplying the homogeneous matrices, creating a function which relates joint and task space.

$$\boldsymbol{T}_{J6/B}(\boldsymbol{\theta}) = \boldsymbol{T}_{J1/B}(\theta_1)\ \boldsymbol{T}_{J2/J1}(\theta_2)\ \boldsymbol{T}_{J3/J2}(\theta_3)\ \boldsymbol{T}_{J4/J3}(\theta_4)\ \boldsymbol{T}_{J5/J4}(\theta_5)\ \boldsymbol{T}_{J6/J5}(\theta_6) \tag{2.1}$$

Recalling the single components:

$$\boldsymbol{T}_{J1/B}(\theta_1) = \begin{bmatrix} \boldsymbol{R}_z(\theta_1) & \boldsymbol{l}_{J1} \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix}, \quad \boldsymbol{T}_{J2/J1}(\theta_2) = \begin{bmatrix} \boldsymbol{R}_y(\theta_2) & \boldsymbol{l}_{J2} \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix}$$

$$\boldsymbol{T}_{J3/J2}(\theta_3) = \begin{bmatrix} \boldsymbol{R}_y(-\theta_3) & \boldsymbol{l}_{J3} \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix}, \quad \boldsymbol{T}_{J4/J3}(\theta_4) = \begin{bmatrix} \boldsymbol{R}_x(-\theta_4) & \boldsymbol{l}_{J4} \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix}$$

$$\boldsymbol{T}_{J5/J4}(\theta_5) = \begin{bmatrix} \boldsymbol{R}_y(-\theta_5) & \boldsymbol{l}_{J5} \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix}, \quad \boldsymbol{T}_{J6/J5}(\theta_6) = \begin{bmatrix} \boldsymbol{R}_x(-\theta_6)\boldsymbol{R}_{J5/J6} & \boldsymbol{l}_{J6} \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix}$$

Equation 2.1 shows the FK which describes the internal rotations of GP12, linking the base frame with the *J6* frame. The following step places the robot within the facility reference frame and incorporates the end effector mounting on *J6*, yielding the complete pose of the end effector with respect to the inertial facility frame $\{G\}$:

$$\boldsymbol{T}_{EE/G}(\boldsymbol{\theta}) = \boldsymbol{T}_{B/G}\ \boldsymbol{T}_{J6/B}(\boldsymbol{\theta})\ \boldsymbol{T}_{EE/J6} \tag{2.2}$$

The final FK Equation 2.2, shows how the pose of the end effector can be determined in task space, knowing the joint space solution. It is important to note that the total number of homogeneous matrices is eight: six of them depend on the joint solution, while the remaining two are constants that define the pose of GP12 base in the ground reference

frame and the position of the end effector, mounted on the robot's last joint.

2.1.3. Inverse Kinematics

Generally speaking, the inverse kinematics of a generic robotic manipulator is difficult and often requires numerical methods to solve. However, for manipulators that meet certain criteria, such as the Pieper criteria [25], closed-form solutions can be derived. Like many other industrial 6R manipulator, the GP12 robotic manipulator is one such example, allowing for efficient computation of joint angles given a desired end-effector pose.

The inverse kinematics algorithm for the GP12 manipulator can be summarized in the following steps:

1. **Solving the position problem**
2. **Solving the orientation problem**

This procedure is a direct consequence of the Pieper criteria, which allows to decouple the position and orientation problems when the last 3 joints are revolute and their rotation axes intersect in a single point (in this case, joints 4, 5 and 6). For this specific mechanism, the position of the wrist center (intersection point of joint axes 4, 5 and 6) can be computed by subtracting the end-effector offset from the desired end-effector position along the direction of the end-effector's z-axis.

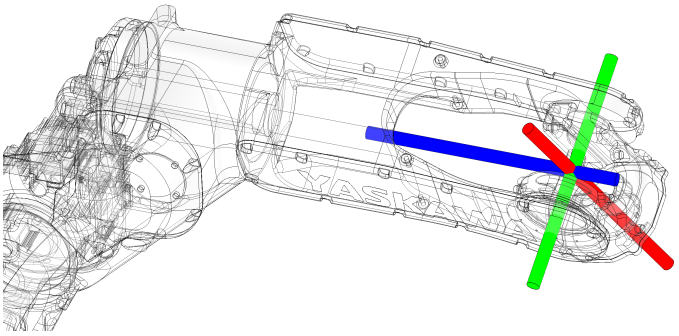


Figure 2.3: Wrist center of GP12.

The wrist center can be seen in Figure 2.3, created by the intersection of ${}^{J4}\hat{\boldsymbol{e}}_1$, ${}^{J5}\hat{\boldsymbol{e}}_2$ and ${}^{J6}\hat{\boldsymbol{e}}_3$: drawn in blue, green and red respectively.

For the IK algorithm, it is useful to remind that the target (or goal) pose ${}^{G}\boldsymbol{T}_{goal}$ is composed by the target position, ${}^{G}\boldsymbol{p}_{goal}$, and the target attitude, ${}^{G}\boldsymbol{R}_{goal}$. These quantities are defined in the *ground* (or *facility*) reference frame $\{G\}$ and ${}^{G}\boldsymbol{R}_{goal}$ defines the $\boldsymbol{R}_{G/EE}$ desired rotation.

Position problem

The first step regards finding the position of the wrist center ${}^{G}\boldsymbol{p}_{wc}$ in the local frame of the robot: this is achieved by subtracting ${}^{G}\boldsymbol{l}_{EE}$ and ${}^{G}\boldsymbol{l}_{J6}$ from the target position ${}^{G}\boldsymbol{p}_{goal}$, along the target attitude.

The following vectors will indicate, via a superscript, the reference frame in which they are defined.

$$
\begin{aligned}
{}^{G}\boldsymbol{p}_{wc} &= {}^{G}\boldsymbol{p}_{goal} - \boldsymbol{R}_{G/J6}\left[{}^{J6}\boldsymbol{l}_{J6} + {}^{J6}\boldsymbol{l}_{EE}\right] \\
{}^{B}\boldsymbol{p}_{wc} &= \boldsymbol{R}_{B/G}\left[{}^{G}\boldsymbol{p}_{wc} - {}^{G}\boldsymbol{p}_{B}\right]
\end{aligned}
$$

All the quantities used in the equation above are known, since the desired end effector's pose (${}^{G}\boldsymbol{T}_{goal}$) is the input of the IK, while ${}^{J6}\boldsymbol{l}_{J6}$ is known from Table 2.1 and ${}^{J6}\boldsymbol{l}_{EE}$ is an end effector's characteristic. ${}^{G}\boldsymbol{p}_{B}$ is used to define the GP12 position in the facility. The matrix $\boldsymbol{R}_{G/J6}$ is easily calculated knowing $\boldsymbol{R}_{G/EE} = \boldsymbol{R}_{G/J6}\boldsymbol{R}_{J6/EE}$, where both $\boldsymbol{R}_{G/EE}$ and $\boldsymbol{R}_{J6/EE}$ are known. Here, $\boldsymbol{R}_{G/EE}$ is derived from the goal homogeneous transformation ${}^{G}\boldsymbol{T}_{goal}$, while $\boldsymbol{R}_{J6/EE}$ is the fixed rotation matrix determined by the mounting of the end effector on joint 6.

It is useful to remind that the matrix $\boldsymbol{R}_{B/G}$ derives from $\boldsymbol{T}_{B/G}$ and it is used to rotate vectors between the *ground* and *base* frames. This rotation matrix is user defined, and corresponds to the base pose of the robot. From now on in the IK section, each position will be expressed in the $\{B\}$ frame of the robot, omitting the superscript, e.g., ${}^{B}\boldsymbol{p}_{wc}$ is written as $\boldsymbol{p}_{wc}$.

Theta 1 The first angle: θ_1 is the azimuth of $\boldsymbol{p}_{wc}$. Figure 2.4 shows a top view of the robotic manipulator.

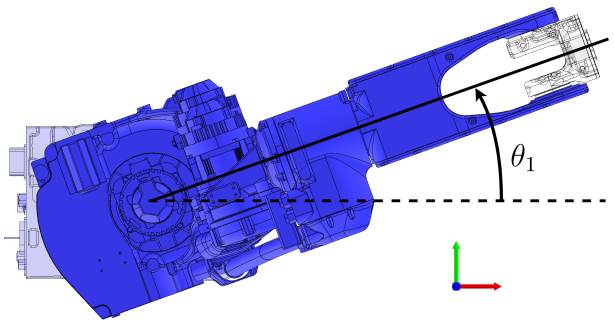


Figure 2.4: GP12 top view, showing θ_1 deflection.

The θ_1 equation uses the MATLAB function *atan2* to evaluate the azimuth of $\boldsymbol{p}_{wc}$. Equation 2.3 shows the implementation:

$$\theta_1 = \text{Az}_{\boldsymbol{p}_{wc}} = \text{atan2}(\boldsymbol{p}^y_{wc}, \boldsymbol{p}^x_{wc}) \tag{2.3}$$

Theta 2 and theta 3 θ_2 and θ_3 are calculated using trigonometry.

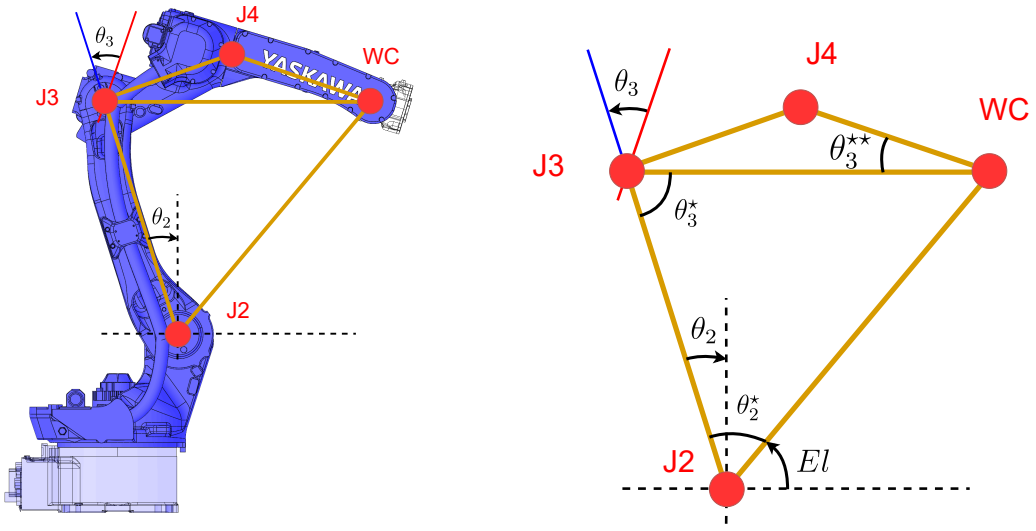


Figure 2.5: GP12 θ_2 and θ_3.

In Figure 2.5 are represented the joint positions: *J2*, *J3*, *J4* and *WC*. The distances between these joints are called a_1, a_2 and a_3 respectively for the *J2–J3* link, the *J3–WC* link and the *J2–WC* fictitious link. It is important to note how these quantities are scalar, and their value is determined from Table 2.1 for the first two, while the latter is calculated as the distance between the new *J2* position (found with FK, using θ_1) and $\boldsymbol{p}_{wc}$.

The first quantity to be evaluated is the elevation of $\boldsymbol{p}_{wc}$:

$$\begin{cases} \boldsymbol{p}_{J2-wc} = \boldsymbol{p}_{wc} - \boldsymbol{p}_{J2} \\ El = \text{atan2}\left(\boldsymbol{p}^z_{J2-wc}, \text{sign}\left(\boldsymbol{p}^x_{J2-wc}\right)\sqrt{{\boldsymbol{p}^x_{J2-wc}}^2 + {\boldsymbol{p}^y_{J2-wc}}^2}\right) \end{cases}$$

The second argument of the *atan2* function is the projection, with sign, of $\boldsymbol{p}_{wc}$ on the xy-plane.

$\theta_2^{\star}$ and $\theta_3^{\star}$ are now introduced as two auxiliary variables; they are used to solve for θ_2

and θ_3. Regarding these auxiliary angles, their values are found by solving the following system of equations:

$$\begin{cases} a_1 \cdot \sin(\theta_2^\star) = a_2 \cdot \sin(\theta_{J2-WC-J3}) \\ a_2^2 = a_1^2 + a_3^2 - 2 \cdot a_1 \cdot a_3 \cdot \cos(\theta_2^\star) \\ \theta_2^\star + \theta_3^\star + \theta_{J2-WC-J3} = \pi \end{cases}$$

The system can be solved, remembering that $0 \leq \theta_2^\star < \pi/2$ and $0 \leq \theta_{J2-WC-J3} < \pi$ for construction and characteristics of the system.

Continuing with θ_2 calculations, its value is found as:

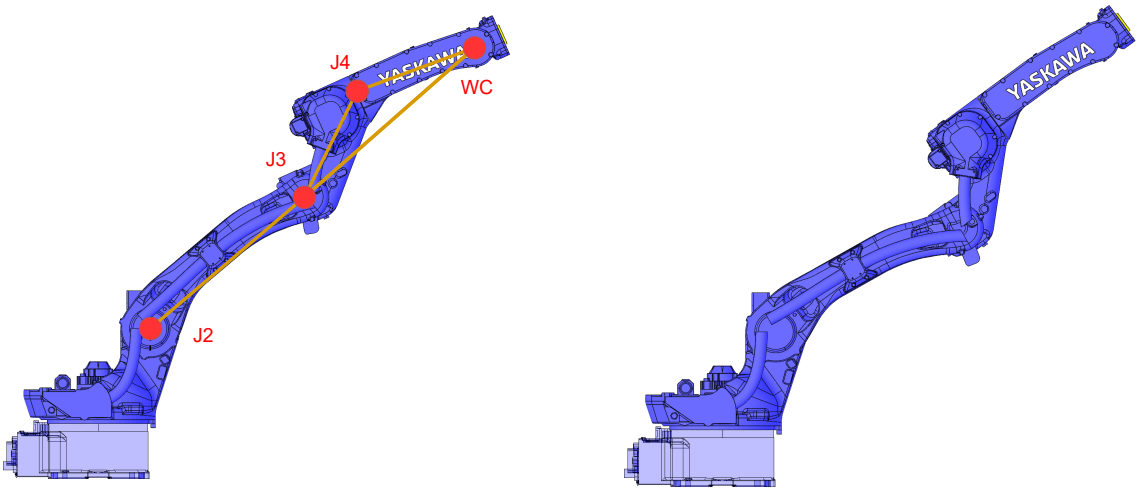
$$\theta_2 = \frac{\pi}{2} - \theta_2^\star - El \tag{2.4}$$

Concerning θ_3, an additional angle must be defined: the angle *J4–WC–J3*, called $\theta_3^{\star\star}$. This last angle is easily calculated via trigonometry, since the *J3–J4–WC* triangle is fully defined.

$$\theta_3 = -\left(\frac{\pi}{2} + \theta_3^{\star\star} - \theta_3^\star\right) \tag{2.5}$$

Equation 2.5 completes the first part of the IK. It is now possible to test this algorithm with the FK, in order to check the position of the wrist center. This procedure allow also to obtain $\boldsymbol{T}_{B/J4}$, which will be useful in the next step.

One final and important note regards the implementation of the *Position problem*: it was chosen a trigonometric approach for its simplicity, leading to Equation 2.4 and Equation 2.4 which are solved using scalar quantities. This procedure is not valid for certain edge-cases, where the *J2–J3–WC* triangle may assume different shapes: e.g., when *J2–J3–WC* align and θ_3 keeps increasing. Those cases have been considered as not of interest, since they are extremely close to the maximum values allowable for J3. Figure 2.6 shows the limit configuration for θ_3, where the elbow shifts from the *up* to the *down* configuration (shown on the right).



θ_3 limit Elbow-down configuration

Figure 2.6: GP12 IK limits.

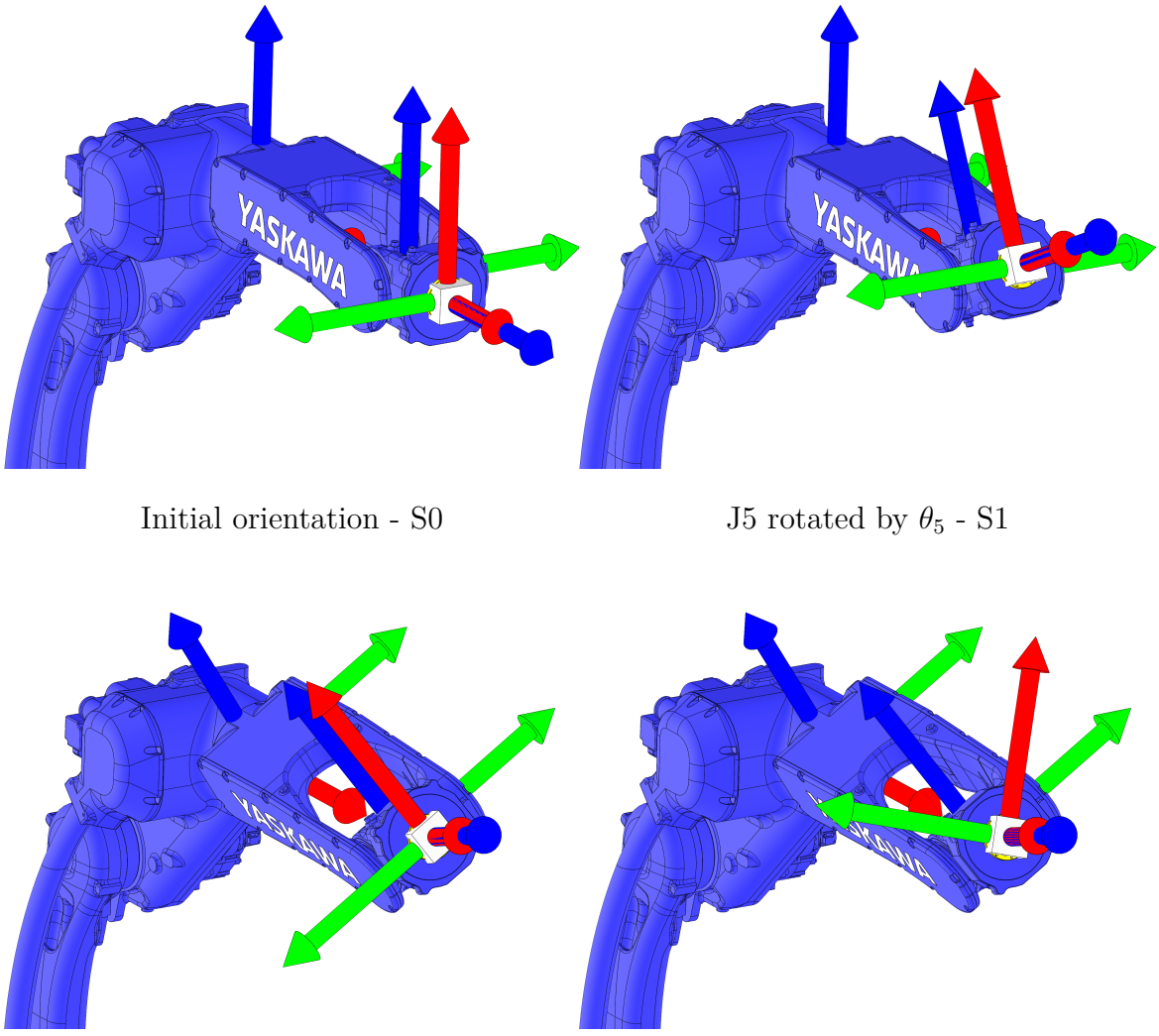
The *J2–J3–WC* aligned configuration, which correspond to the maximum reach solution, can be seen in the left picture, while the right figure represents the elbow-down configuration.

Orientation problem

Before starting with the *orientation problem*, the non singular solution of the last three joints must be solved. The issue involves two distinct behaviors of the joints: following this paragraph will be clear how, different combinations of θ_5 and θ_4 will produce the same result, while, θ_6, shows a 2π periodicity of the solution. This is solved by reducing the θ_5 solution to rotate joint B in the B+ direction and θ_6 to rotate joint T in the T- direction. These choices are completely arbitrary, and any other convention of rotation of the last three joints can be used: the IK algorithm uses an input to determine which condition is applied, leading to different behaviors.

The orientation problem aims at determining the solution of the last three joints of GP12, in order to match ${}^{G}\boldsymbol{R}_{goal}$ with $\boldsymbol{R}_{G/EE}$. It is important noting that the position will automatically be matched as a consequence of the wrist center match (solved by the *Position problem*) and, now, the orientation match.

The three phases are reported below, where the figures represent each step: θ_5, θ_4 and θ_6 will be found in this order.



Initial orientation - S0 J5 rotated by θ_5 - S1

J4 rotated by θ_4 - S2 J6 rotated by θ_6 - S3

Figure 2.7: GP12 - Orientation problem steps.

For each step reported in figure, a label is applied: S0, S1, S2 and S3. This will bound each frame or axis to a specific step: e.g., ${}^{J4}\hat{\boldsymbol{e}}_1^{S0}$ is the x-axis of the forth joint (J4, controlled by θ_4), in the *Initial orientation*, as reported in Figure 2.7. With *Initial orientation* referring to the FK applied to the first three joints (solution found in the *position problem*), while leaving a zero deflection to the last three joints.

It can be seen in the S0 step of Figure 2.7 how all frames of J4, J5 and J6 joints are aligned: this is always true at this step of the IK, since only the first three angles have been modified.

Theta 5 θ_5 is selected such that the angle between ${}^{J6}\hat{\boldsymbol{e}}_3^{S1}$ and ${}^{J6}\hat{\boldsymbol{e}}_3^{S0}$ coincides with the angle between ${}^{G}\hat{\boldsymbol{e}}_1^{goal}$ and ${}^{J6}\hat{\boldsymbol{e}}_3^{S0}$.

$$\theta_5 = \text{atan2}\left(\frac{||{}^{J6}\hat{\boldsymbol{e}}_3^{S0} \times {}^{G}\hat{\boldsymbol{e}}_1^{goal}||}{||{}^{J6}\hat{\boldsymbol{e}}_3^{S0}||\ ||{}^{G}\hat{\boldsymbol{e}}_1^{goal}||}, \frac{{}^{J6}\hat{\boldsymbol{e}}_3^{S0} \cdot {}^{G}\hat{\boldsymbol{e}}_1^{goal}}{||{}^{J6}\hat{\boldsymbol{e}}_3^{S0}||\ ||{}^{G}\hat{\boldsymbol{e}}_1^{goal}||}\right) \tag{2.6}$$

Although these angles now equal to each other, the corresponding directions are not yet aligned; this is expected and will be corrected in the next step.

Theta 4 The purpose of θ_4 is to align ${}^{J6}\hat{\boldsymbol{e}}_3^{S1}$ with ${}^{G}\hat{\boldsymbol{e}}_1^{goal}$, producing ${}^{J6}\hat{\boldsymbol{e}}_3^{S2}$ which is equal to ${}^{J6}\hat{\boldsymbol{e}}_3^{S3}$ and to ${}^{G}\hat{\boldsymbol{e}}_1^{goal}$. This is achieved by rotating the vector ${}^{J6}\hat{\boldsymbol{e}}_3^{S1}$ around ${}^{J4}\hat{\boldsymbol{e}}_1^{S1}$, by imposing the deflection angle θ_4.

This procedure requires the definition of two auxiliary vectors, that are perpendicular to ${}^{J4}\hat{\boldsymbol{e}}_1^{S1}$ by construction, defined as:

$$\boldsymbol{k}_1^{S1} = {}^{J6}\hat{\boldsymbol{e}}_3^{S1} \times {}^{J4}\hat{\boldsymbol{e}}_1^{S1} \quad \text{and} \quad \boldsymbol{k}_2^{S1} = {}^{G}\hat{\boldsymbol{e}}_1^{goal} \times {}^{J4}\hat{\boldsymbol{e}}_1^{S1}$$

Finally, the equation for θ_4 is the following:

$$\theta_4 = -\text{atan2}\left(S_{\theta_4}\frac{||\boldsymbol{k}_1^{S1} \times \boldsymbol{k}_2^{S1}||}{||\boldsymbol{k}_1^{S1}||\ ||\boldsymbol{k}_2^{S1}||}, \frac{\boldsymbol{k}_1^{S1} \cdot \boldsymbol{k}_2^{S1}}{||\boldsymbol{k}_1^{S1}||\ ||\boldsymbol{k}_2^{S1}||}\right) \tag{2.7}$$

With $S_{\theta_4} = \text{sign}\left[(\boldsymbol{k}_1^{S1} \times \boldsymbol{k}_2^{S1}) \cdot {}^{J4}\hat{\boldsymbol{e}}_1^{S1}\right]$ defining the sign of the first argument for the *atan2* function evaluation.

Theta 6 The final angle is θ_6: this rotation around $\mathbf{T}$, aligns $-{}^{J6}\hat{\boldsymbol{e}}_2^{S2}$ with ${}^{G}\hat{\boldsymbol{e}}_2^{goal}$, and ${}^{J6}\hat{\boldsymbol{e}}_1^{S2}$ with ${}^{G}\hat{\boldsymbol{e}}_3^{goal}$, producing the final $\boldsymbol{R}_{G/J6}$ which is identical to ${}^{G}\boldsymbol{R}_{goal}$.

$$\theta_6 = -\text{atan2}\left(S_{\theta_6}\frac{||-{}^{J6}\hat{\boldsymbol{e}}_2^{S2} \times {}^{G}\hat{\boldsymbol{e}}_2^{goal}||}{||-{}^{J6}\hat{\boldsymbol{e}}_2^{S2}||\ ||{}^{G}\hat{\boldsymbol{e}}_2^{goal}||}, \frac{-{}^{J6}\hat{\boldsymbol{e}}_2^{S2} \cdot {}^{G}\hat{\boldsymbol{e}}_2^{goal}}{||-{}^{J6}\hat{\boldsymbol{e}}_2^{S2}||\ ||{}^{G}\hat{\boldsymbol{e}}_2^{goal}||}\right) \tag{2.8}$$

With $S_{\theta_6} = \text{sign}\left[(-{}^{J6}\hat{\boldsymbol{e}}_2^{S2} \times {}^{G}\hat{\boldsymbol{e}}_2^{goal}) \cdot {}^{J6}\hat{\boldsymbol{e}}_3^{S2}\right]$ defining the sign of the first argument for the *atan2* function evaluation.

The assumptions made to solve the *multiple solution* problem are critical for a robust implementation of the IK, since the algorithm shall be able to provide a continuous solution for a point-like trajectory of ${}^{G}\boldsymbol{R}_{goal}$, without sudden jumps in the joint angles. It is worth noting that an internal IK optimization can be adopted (to produce continuous optimized solutions), but it would break the continuity of similar solutions in the case of an optimization, which is problematic for gradient-based methods.

On the other side, the optimal joint solution, in terms of smaller joint deflection, was implemented as it is useful when dealing with isolated end-effector poses, where continuity is not required. This approach is not used in the later discussed facility optimization, but it is useful for testing purposes.

The IK solver provides a solution even for unreachable wrist-center points, employing the limit reachable configuration. It also outputs a signed reachability margin (negative if reachable, positive otherwise), which is used as a nonlinear constraint for the optimization.

2.1.4. Velocity Kinematics

The Velocity Kinematics (VK) maps velocities of the end effector (in task space) to velocities of the joints (in joint space).

For a minimal set of coordinates $\boldsymbol{x} \in \mathbb{R}^m$, the velocities are given by $\dot{\boldsymbol{x}} = d\boldsymbol{x}/dt \in \mathbb{R}^m$. For this problem, the FK can be defined as $\boldsymbol{x}(t) = f(\boldsymbol{\theta}(t))$, whose pose of the end effector can be retrieved knowing the joint solution. By differentiating with the chain rule, the following equation can be retrieved [17]:

$$\dot{\boldsymbol{x}} = \frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \frac{d\boldsymbol{\theta}(t)}{dt} = \frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \dot{\boldsymbol{\theta}} = \boldsymbol{J}(\boldsymbol{\theta}) \dot{\boldsymbol{\theta}} \tag{2.9}$$

where $\boldsymbol{J}(\boldsymbol{\theta})$ is called the Jacobian. This relates the end effector velocity $\dot{\boldsymbol{x}}$ with the joint velocity $\dot{\boldsymbol{\theta}}$.

To fully describe the end effector motion in space, the Twist is now introduced. Denoted with $\boldsymbol{\nu} \in \mathbb{R}^6$, it is composed by the angular velocity $\boldsymbol{\omega}$ and the linear velocity $\boldsymbol{v}$:

$$\boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} \in \mathbb{R}^6$$

Returning to the VK, the twist is related to the joint velocities by

$$\boldsymbol{\nu} = \boldsymbol{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}$$

This equation can be inverted to solve for joint velocities:

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}(\boldsymbol{\theta})^{-1}\boldsymbol{\nu} \tag{2.10}$$

In Equation 2.10, the Jacobian matrix is square and invertible, while the $\dot{\boldsymbol{\theta}} = \boldsymbol{J}(\boldsymbol{\theta})^{\dagger}\boldsymbol{\nu}$ uses the Moore-Penrose pseudoinverse (denoted by the dagger symbol) for cases where the Jacobian is not square or singular. This last method is more general and can handle a wider range of scenarios, particularly cases where the Jacobian may not be square due to redundant degrees of freedom. Furthermore, Equation 2.10 is the inverse VK problem, called Inverse Velocity Kinematics (IVK).

It is worth remaking that both Jacobian and Twist used from now on are expressed in the fixed *space* frame, and not in the *body* frame [17].

Jacobian

The Jacobian matrix, defined in Equation 2.9, is created by partially differentiating the FK by the joint angles $\boldsymbol{\theta}$.

$$\boldsymbol{J}(\boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{J}_{\omega}(\boldsymbol{\theta}) \\ \boldsymbol{J}_{v}(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} \dfrac{\partial \boldsymbol{\omega}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \\ \dfrac{\partial \boldsymbol{v}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \omega_x}{\partial \theta_1} & \frac{\partial \omega_x}{\partial \theta_2} & \frac{\partial \omega_x}{\partial \theta_3} & \frac{\partial \omega_x}{\partial \theta_4} & \frac{\partial \omega_x}{\partial \theta_5} & \frac{\partial \omega_x}{\partial \theta_6} \\ \frac{\partial \omega_y}{\partial \theta_1} & \frac{\partial \omega_y}{\partial \theta_2} & \frac{\partial \omega_y}{\partial \theta_3} & \frac{\partial \omega_y}{\partial \theta_4} & \frac{\partial \omega_y}{\partial \theta_5} & \frac{\partial \omega_y}{\partial \theta_6} \\ \frac{\partial \omega_z}{\partial \theta_1} & \frac{\partial \omega_z}{\partial \theta_2} & \frac{\partial \omega_z}{\partial \theta_3} & \frac{\partial \omega_z}{\partial \theta_4} & \frac{\partial \omega_z}{\partial \theta_5} & \frac{\partial \omega_z}{\partial \theta_6} \\ \frac{\partial v_x}{\partial \theta_1} & \frac{\partial v_x}{\partial \theta_2} & \frac{\partial v_x}{\partial \theta_3} & \frac{\partial v_x}{\partial \theta_4} & \frac{\partial v_x}{\partial \theta_5} & \frac{\partial v_x}{\partial \theta_6} \\ \frac{\partial v_y}{\partial \theta_1} & \frac{\partial v_y}{\partial \theta_2} & \frac{\partial v_y}{\partial \theta_3} & \frac{\partial v_y}{\partial \theta_4} & \frac{\partial v_y}{\partial \theta_5} & \frac{\partial v_y}{\partial \theta_6} \\ \frac{\partial v_z}{\partial \theta_1} & \frac{\partial v_z}{\partial \theta_2} & \frac{\partial v_z}{\partial \theta_3} & \frac{\partial v_z}{\partial \theta_4} & \frac{\partial v_z}{\partial \theta_5} & \frac{\partial v_z}{\partial \theta_6} \end{bmatrix} \tag{2.11}$$

Equation 2.11 shows the components of the Jacobian matrix: the first three rows are created with the angular velocities contributions, while the last three rows are created with linear velocities contributions.

There are several ways to derive the Jacobian matrix of open chains systems, such as the GP12:

1. Using MATLAB Robotic Toolbox functions;
2. Employing symbolic or automatic differentiation to define the FK and analytically compute each component of the Jacobian (e.g., via MATLAB Symbolic Math Toolbox[3]);

[3]MathWorks, Symbolic Math Toolbox. Available at `https://mathworks.com/products/symbolic.html`

3. Using joint axes to compute directly the *Geometric Jacobian* in the Space Frame [17, 28].

The best method imply calculating the Jacobian using the joint axes, and then using the Symbolic Math Toolbox to define an anonymous function: this procedure is fast both in the Jacobian construction and in its evaluation.

The procedure require the calculation of rotation axis and position of each joint of the robot. These values are then combined to create the final Jacobian:

$$\boldsymbol{J}(\boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{J}_\omega(\boldsymbol{\theta}) \\ \boldsymbol{J}_v(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega}_{s1}(\boldsymbol{\theta}) & \boldsymbol{\omega}_{s2}(\boldsymbol{\theta}) & \boldsymbol{\omega}_{s3}(\boldsymbol{\theta}) & \boldsymbol{\omega}_{s4}(\boldsymbol{\theta}) & \boldsymbol{\omega}_{s5}(\boldsymbol{\theta}) & \boldsymbol{\omega}_{s6}(\boldsymbol{\theta}) \\ \boldsymbol{v}_{s1}(\boldsymbol{\theta}) & \boldsymbol{v}_{s2}(\boldsymbol{\theta}) & \boldsymbol{v}_{s3}(\boldsymbol{\theta}) & \boldsymbol{v}_{s4}(\boldsymbol{\theta}) & \boldsymbol{v}_{s5}(\boldsymbol{\theta}) & \boldsymbol{v}_{s6}(\boldsymbol{\theta}) \end{bmatrix} \tag{2.12}$$

where each rotation axis $\boldsymbol{\omega}_s$ and position $\boldsymbol{v}_s$ are defined in the $\{G\}$ frame and explicitly dependant on joint solutions. For a revolute joint, the linear velocity component of the spatial twist, $\boldsymbol{v}_s$, is given by:

$$\boldsymbol{v}_{si}(\boldsymbol{\theta}) = \boldsymbol{\omega}_{si}(\boldsymbol{\theta}) \times [\boldsymbol{q}_{EE}(\boldsymbol{\theta}) - \boldsymbol{q}_i(\boldsymbol{\theta})], \quad \text{for } \; i = 1, \ldots, n \tag{2.13}$$

where n is the total number of joints of the robot.

Analyzing now each element, the joints rotation matrices are the same used in section 2.1.2, while the rotation axes and positions are calculated as reported below.

$$\begin{aligned} \boldsymbol{q}_1 &= {}^G\boldsymbol{p}_{base} + \boldsymbol{R}_{G/B}\, \boldsymbol{l}_1 & \boldsymbol{\omega}_{s1} &= \boldsymbol{R}_{G/B}\, [0\ 0\ 1]^T \\ \boldsymbol{q}_2(\boldsymbol{\theta}) &= \boldsymbol{q}_1 + \boldsymbol{R}_{G/J1}(\boldsymbol{\theta})\, \boldsymbol{l}_2 & \boldsymbol{\omega}_{s2}(\boldsymbol{\theta}) &= \boldsymbol{R}_{G/J1}(\boldsymbol{\theta})\, [0\ 1\ 0]^T \\ \boldsymbol{q}_3(\boldsymbol{\theta}) &= \boldsymbol{q}_2(\boldsymbol{\theta}) + \boldsymbol{R}_{G/J2}(\boldsymbol{\theta})\, \boldsymbol{l}_3 & \boldsymbol{\omega}_{s3}(\boldsymbol{\theta}) &= \boldsymbol{R}_{G/J2}(\boldsymbol{\theta})\, [0\ -1\ 0]^T \\ \boldsymbol{q}_4(\boldsymbol{\theta}) &= \boldsymbol{q}_3(\boldsymbol{\theta}) + \boldsymbol{R}_{G/J3}(\boldsymbol{\theta})\, \boldsymbol{l}_4 & \boldsymbol{\omega}_{s4}(\boldsymbol{\theta}) &= \boldsymbol{R}_{G/J3}(\boldsymbol{\theta})\, [-1\ 0\ 0]^T \\ \boldsymbol{q}_5(\boldsymbol{\theta}) &= \boldsymbol{q}_4(\boldsymbol{\theta}) + \boldsymbol{R}_{G/J4}(\boldsymbol{\theta})\, \boldsymbol{l}_5 & \boldsymbol{\omega}_{s5}(\boldsymbol{\theta}) &= \boldsymbol{R}_{G/J4}(\boldsymbol{\theta})\, [0\ -1\ 0]^T \\ \boldsymbol{q}_6(\boldsymbol{\theta}) &= \boldsymbol{q}_5(\boldsymbol{\theta}) + \boldsymbol{R}_{G/J5}(\boldsymbol{\theta})\, \boldsymbol{l}_6 & \boldsymbol{\omega}_{s6}(\boldsymbol{\theta}) &= \boldsymbol{R}_{G/J5}(\boldsymbol{\theta})\, [-1\ 0\ 0]^T \end{aligned}$$

Regarding Equation 2.13, $\boldsymbol{q}_{EE}(\boldsymbol{\theta}) = \boldsymbol{q}_6(\boldsymbol{\theta}) + \boldsymbol{R}_{G/EE}(\boldsymbol{\theta})\, \boldsymbol{l}_{EE}$. Otherwise, if the end effector is not mounted, $\boldsymbol{q}_{EE}(\boldsymbol{\theta})$ is exactly $\boldsymbol{q}_6(\boldsymbol{\theta})$.

2.1.5. Manipulability and singularity analysis

Robotic manipulators are affected by singularities: configurations where the dynamics of pre-determined trajectories may be inaccurate, due to quick joints acceleration or rapid

change of configuration. These points of singularities have to be avoided in the trajectory mapping process for the facility, in order to correctly reproduce the designed trajectories and prevent joint saturations.

Lynch at al. [17] identified a parameter of merit for the singularity proximity analysis of a robotic manipulator: the *manipulability ellipsoid*. This geometrical shape is defined for a generic n-joint open chain, where the task space coordinates are defined by $\boldsymbol{q} \in \mathbb{R}^m$, $m \leq n$.

The following equation reconstructs the ellipsoid of the end effector velocities for joint rates $\dot{\boldsymbol{\theta}}$ which satisfy $||\dot{\boldsymbol{\theta}}|| = 1$:

$$\begin{aligned} 1 &= \dot{\boldsymbol{\theta}}^T \dot{\boldsymbol{\theta}} \\ &= \left(\boldsymbol{J}^{-1}\dot{\boldsymbol{q}}\right)^T \left(\boldsymbol{J}^{-1}\dot{\boldsymbol{q}}\right) \\ &= \dot{\boldsymbol{q}}^T \boldsymbol{J}^{-T}\boldsymbol{J}^{-1}\dot{\boldsymbol{q}} \\ &= \dot{\boldsymbol{q}}^T \left(\boldsymbol{J}\boldsymbol{J}^T\right)^{-1} \dot{\boldsymbol{q}} \\ &= \dot{\boldsymbol{q}}^T \boldsymbol{A}^{-1}\dot{\boldsymbol{q}} \end{aligned}$$

So, the equation of the manipulability ellipsoid is:

$$\dot{\boldsymbol{q}}^T \boldsymbol{A}^{-1}\dot{\boldsymbol{q}} = 1, \quad \text{where} \quad \boldsymbol{A} = \boldsymbol{J}\boldsymbol{J}^T$$

where $\boldsymbol{J} = \boldsymbol{J}(\boldsymbol{\theta})$ and $\boldsymbol{A} = \boldsymbol{A}(\boldsymbol{\theta})$.

Going now back to the standard formulation, the Jacobian can be split into two different sections: the upper part defines the angular velocities, while the lower part defines the linear velocities, as reported in Equation 2.11. This allow to construct two different $\boldsymbol{A}(\boldsymbol{\theta})$ matrices:

$$\begin{aligned} \boldsymbol{A}_\omega(\boldsymbol{\theta}) &= \boldsymbol{J}_\omega(\boldsymbol{\theta})\boldsymbol{J}_\omega^T(\boldsymbol{\theta}) \\ \boldsymbol{A}_v(\boldsymbol{\theta}) &= \boldsymbol{J}_v(\boldsymbol{\theta})\boldsymbol{J}_v^T(\boldsymbol{\theta}) \end{aligned}$$

It can be noted how the axes of the ellipsoid are defined by the eigenvalues of the matrix $\boldsymbol{A}$, since $\dot{\boldsymbol{q}}^T \boldsymbol{A}^{-1}\dot{\boldsymbol{q}} = 1$ correspond to a quadratic formulation. The size of the ellipsoid is representative of the proximity to singularities. In this example, the eigenvalues are a total of three for each type of $\boldsymbol{A}(\boldsymbol{\theta})$ matrix. The final shape of the ellipsoid is directly linked to the singularity characteristics of the specific $\boldsymbol{\theta}$ configuration.

If the shape of the ellipsoid is not symmetric, then the direction corresponding to the minor

axis indicates the weakest direction of movement of the robot. If this values collapses close to zero, then this represent a singularity. For this reason, the relations between the eigenvalues can be used as a singularity indicator:

$$\begin{aligned} \mu_1(\boldsymbol{A}) &= \sqrt{\frac{\lambda_{max}(\boldsymbol{A})}{\lambda_{min}(\boldsymbol{A})}} \geq 1 \\ \mu_2(\boldsymbol{A}) &= \frac{\lambda_{max}(\boldsymbol{A})}{\lambda_{min}(\boldsymbol{A})} \geq 1 \\ \mu_3(\boldsymbol{A}) &= \sqrt{\lambda_1 \lambda_2 \lambda_3} = \sqrt{\det(\boldsymbol{A})} \end{aligned} \tag{2.14}$$

Equation 2.14 shows three typical indicators [17], for a given $\boldsymbol{A} = \boldsymbol{A}(\boldsymbol{\theta})$ matrix.

Starting from μ_1, it describes the ratio between the maximum and the minimum eigenvalues of the matrix A, which can be seen as the ratio between the greater and smaller ellipsoid axes, with a final value greater than or equal to 1. A value close to 1 indicates a quasi spherical ellipsoid, while larger values indicate a great difference between the axis, indicating a singularity. The same applies to μ_2, which is simply the square of μ_1. Instead, μ_3, represents the volume of the ellipsoid, which is directly proportional to the manipulability of the robot: larger volumes indicate better manipulability, while smaller volumes indicate poor manipulability and proximity to singularities. Such indicators could serve as useful features in the cost function to avoid singularities during the mapping process.

2.2. Facility

First important modification of Table 1.10 consists in defining the new joint limits. These values have been selected by DART Lab, for collision avoidance purposes.

GP12$_1$	**Axis**	**Specification**
Joints range	S-Axis	$[-170°, 170°]$
	L-Axis	$[-90°, 155°]$
	U-Axis	$[-85°, 150°]$
	R-Axis	$[-200°, 200°]$
	B-Axis	$[-140°, 140°]$
	T-Axis	$[-455°, 455°]$

Table 2.2: GP12$_1$ RAFFAELLO joints range.

GP12$_2$	**Axis**	**Specification**
Joints range	S-Axis	$[-170°, 170°]$
	L-Axis	$[-90°, 155°]$
	U-Axis	$[-85°, 150°]$
	R-Axis	$[-200°, 200°]$
	B-Axis	$[-90°, 90°]$
	T-Axis	$[-455°, 455°]$

Table 2.3: GP12$_2$ RAFFAELLO joints range.

Regarding the floor plan of the facility, the two robots are facing each other: the top and side views of the facility can be reconstructed as shown in Figure 2.8 and in Figure 2.9.

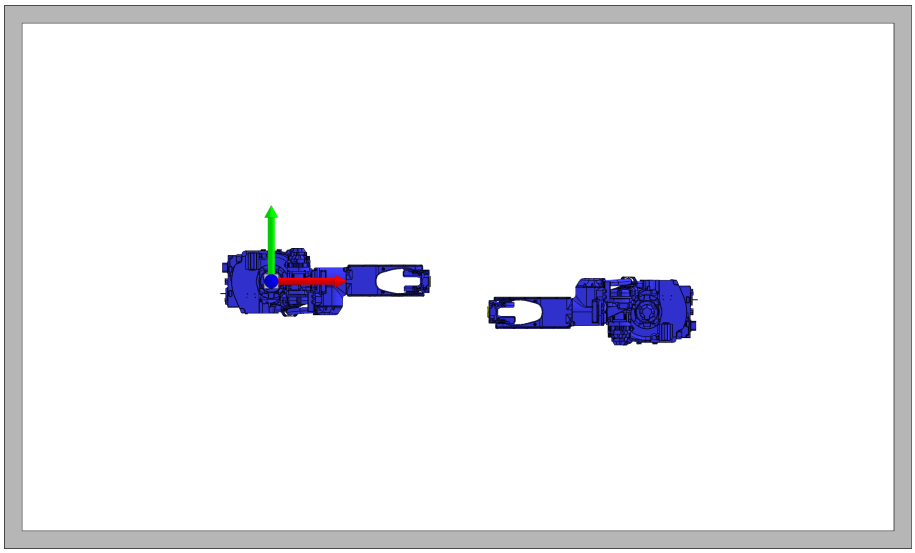


Figure 2.8: RAFFAELLO facility, top view.

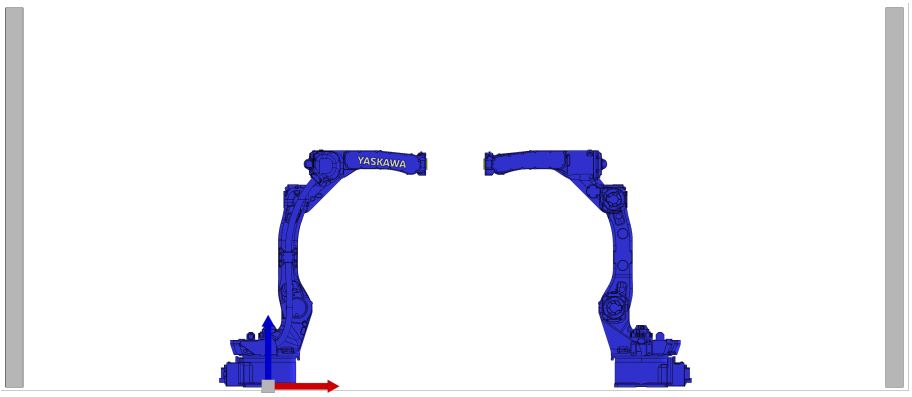


Figure 2.9: RAFFAELLO facility, side view.

The frame $\{G\}$ is reported in the previous figures: denoting with red, green and blue colors the x, y and z axis respectively. This frame is centered in the base of the $GP12_1$. The positions of the bases of the two robots are reported in the following table:

Element	Description	Value [m]
$GP12_1$	x coordinate	0
	y coordinate	0
	z coordinate	0
$GP12_2$	x coordinate	2.1118
	y coordinate	-0.1710
	z coordinate	0.0132

Table 2.4: $GP12_1$ and $GP12_2$ base positions in the $\{G\}$ frame.

Instead, the dimensions of the facility in the frame $\{G\}$ are reported in Table 2.5:

Element	Description	Value [m]
Floor height	z coordinate	0.10
Ceiling height	z coordinate	2.25
Final room height	usable height	2.12
West wall	x coordinate	3.50
South wall	y coordinate	1.45
North wall	y coordinate	-1.75
East wall	x coordinate	-1.40

Table 2.5: Geometrical boundaries of the RAFFAELLO testbed room.

Regarding the communication with the robots, using Yaskawa's MotoCom SDK[4] (Windows DLLs), MATLAB sends individual poses to the GP12 robot controller (YRC1000[5]) for point-to-point command execution.

2.3. Facility Optimization

This section defines and analyses the mapping from real spacecraft trajectories to the trajectories that can be executed by the robotic testbed. Prior robotic knowledge will

[4]YASKAWA, MotoCom SDK. Available at `https://www.yaskawa.it/software/productdetail/product/motocom-sdk_1679`

[5]YASKAWA, YRC1000 controller. Available at `https://www.yaskawa.eu.com/robot/controller/productdetail/product/yrc1000_583`

be used for this scope and HIL will be discussed. V&V will assess the robustness of the methodologies and real facility tests will finally test the implementation.

Starting from the analysis of the system, it is useful to define the *complete system* as the set of all the quantities of the problem: the pose of the target body, the pose of the spacecraft and the Sun direction, all evolving over time.

This *complete system* can be seen as a discrete and frozen cloud of points, vectors and reference frames. Indeed, the reality of HIL imposes the discretization constraints, since the real world applications do not allow for continuous time systems to be easily evaluated and tested, surely not in this field.

Given this, the problem can be analyzed from a geometrical viewpoint, discarding the specific dynamic nature of the system and focusing on a discretized cloud which comprises all the states of the system at different times.

The definition of the "cloud" is the first aspect to be addressed: the main characteristics to be defined are the illumination of the target, the relative motion between spacecraft and target, and the orientation of the spacecraft in space. Starting from an inertial reference frame, centered in the center of mass of the target, the attitude of both target body and spacecraft can be defined via rotation matrices. The position and velocity of the spacecraft, as well as the Sun direction, can instead be defined via vectors.

Regarding the physical constraints of the "cloud", it can be noted that the relative motion between the spacecraft and the target body is inseparable and cannot be altered. The Sun illumination, on the other hand, is defined only by its direction: if the complete system rotates around this Sun direction, the overall system mechanics remain unaltered. Similarly, if the *complete system* translates along the Sun direction, apart from the intensity of the Sun irradiance, the whole system would not change. These two quantities are then selected for the optimization: rotation and translation, of the *cloud*, along the Sun direction.

The optimization will use the vector $\boldsymbol{x}$ for the solution of the problem, denoting with x_1 and x_2 the rotation and the angular velocity of the cloud, and with x_3 the translation.

2.3.1. RAFFAELLO adaptation

The RAFFAELLO facility can achieve arbitrary orientations in $SO(3)$ of both target body and spacecraft, if robotically feasible. The main limitation comes from the lamp used to emulate the Sun direction. Since it is mounted on a fixed tripod, its position cannot be changed during the test. This characteristic of the facility, allow to define the Sun

direction as static, while the *complete system* would ideally be free to rotate and translate around this direction. This mechanics is the core concept of the map optimization and rely on geometric manipulation of the *cloud*.

In order to produce this rigid rotation of the vectors, around the Sun direction, the *Euler-Rodrigues* [9] formula is employed:

$$\boldsymbol{r}_{rot} = \boldsymbol{r}\cos\psi + (\hat{\boldsymbol{k}} \times \boldsymbol{r})\sin\psi + \hat{\boldsymbol{k}}(\hat{\boldsymbol{k}} \cdot \boldsymbol{r})(1 - \cos\psi) \tag{2.15}$$

In Equation 2.15, a generic vector $\boldsymbol{r}$ is rotated along the $\hat{\boldsymbol{k}}$ versor ($||\hat{\boldsymbol{k}}|| = 1$), producing $\boldsymbol{r}_{rot}$. This equation can be generalized in a matrix like formulation, allowing for faster evaluations:

$$\boldsymbol{r}_{rot} = \boldsymbol{R}_{\hat{\boldsymbol{k}}}(\psi)\ \boldsymbol{r}$$

where $\boldsymbol{R}_{\hat{\boldsymbol{k}}}(\psi)$ is the rotation matrix around the axis defined by the unit vector $\hat{\boldsymbol{k}}$ by an angle ψ, which can be determined from Equation 2.15 as [6, 19]:

$$\boldsymbol{R}_{\hat{\boldsymbol{k}}}(\psi) = \boldsymbol{I} + \sin\psi\ [\hat{\boldsymbol{k}}]_\times + (1 - \cos\psi)\ [\hat{\boldsymbol{k}}]_\times [\hat{\boldsymbol{k}}]_\times \tag{2.16}$$

where $\boldsymbol{I}$ is the 3×3 identity matrix and $[\hat{\boldsymbol{k}}]_\times$ is the skew-symmetric matrix of $\hat{\boldsymbol{k}}$, defined as:

$$[\hat{\boldsymbol{k}}]_\times = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix}$$

A demonstration that the natural constraints are preserved is required to show that the new rotated *cloud* exhibits the same characteristics as the original system.

Demonstrations

This procedure can be considered valid if certain criteria are met, regarding the preservation of the *natural* constraints, while the cloud can translate and rotate along the Sun direction. The following points describe these criteria:

- Preservation of the illumination condition on the target surface;

- Preservation of the relative position between spacecraft and target body attitude.

Preservation of illumination condition In a target-centered reference frame, the Sun's phase angle, defined relative to an observer fixed on the target's surface, can serve as the figure of merit for this demonstration. The same illumination condition is achieved if the phase angle ϕ remain constant with the transformation.

First of all, defining the phase angle as: $\phi = \cos^{-1}(\hat{\boldsymbol{r}} \cdot \hat{\boldsymbol{k}})$. Where $\boldsymbol{r}$ and $\hat{\boldsymbol{k}}$ define respectively the position of the random observer and the Sun direction.

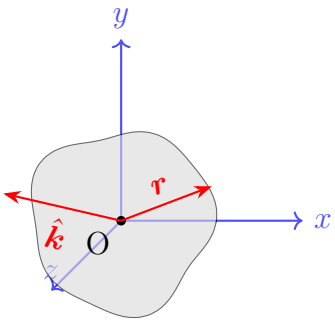


Figure 2.10: Generic body with generic $\boldsymbol{r}$ and $\hat{\boldsymbol{k}}$.

The preservation of the illumination condition is met if

$$\phi = \phi_{rot} \quad \text{or} \quad \boldsymbol{r} \cdot \hat{\boldsymbol{k}} = \boldsymbol{r}_{rot} \cdot \hat{\boldsymbol{k}}_{rot}$$

With new quantities of the rotated system denoted with the subscript *rot*. It can also be noted how $\hat{\boldsymbol{k}}$ and $\hat{\boldsymbol{k}}_{rot}$ are equal (easily found by using Equation 2.15).

This is trivial to prove, as it can be done by evaluating directly the quantities:

$$\hat{\boldsymbol{k}} \cdot \boldsymbol{r}_{rot} = \hat{\boldsymbol{k}} \cdot (\boldsymbol{r} \cos\psi) + \hat{\boldsymbol{k}} \cdot (\hat{\boldsymbol{k}} \times \boldsymbol{r}) \sin\psi + \hat{\boldsymbol{k}} \cdot \left[\hat{\boldsymbol{k}}(\hat{\boldsymbol{k}} \cdot \boldsymbol{r})(1 - \cos\psi)\right]$$

Knowing that $\hat{\boldsymbol{k}} \cdot (\hat{\boldsymbol{k}} \times \boldsymbol{r}) = 0$ and that $\hat{\boldsymbol{k}} \cdot \hat{\boldsymbol{k}} = 1$ (since $\hat{\boldsymbol{k}}$ is a unit vector), we can simplify the equation:

$$\hat{\boldsymbol{k}} \cdot \boldsymbol{r}_{rot} = (\hat{\boldsymbol{k}} \cdot \boldsymbol{r}) \cos\psi + (\hat{\boldsymbol{k}} \cdot \boldsymbol{r})(1 - \cos\psi) = \hat{\boldsymbol{k}} \cdot \boldsymbol{r}$$

This demonstrates that the dot product remains unchanged after the rotation, confirming that the illumination conditions on the target are preserved, for any observer defined by $\boldsymbol{r}$.

Preservation of the relative position In order to demonstrate that the relative position between a trajectory point of the spacecraft and a random observer on the target surface remain unaltered, two vectors $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ can be used to identify these two points, respectively.

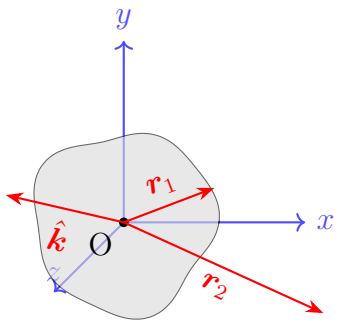


Figure 2.11: Generic body with generic $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ and $\hat{\boldsymbol{k}}$.

The demonstration can be simplified using this matrix form:

$$\begin{aligned}\boldsymbol{r}_{1,rot} \cdot \boldsymbol{r}_{2,rot} &= (\boldsymbol{R}_{\hat{\boldsymbol{k}}}(\psi)\ \boldsymbol{r}_1) \cdot (\boldsymbol{R}_{\hat{\boldsymbol{k}}}(\psi)\ \boldsymbol{r}_2) = \\ &= \boldsymbol{r}_1^T\ \boldsymbol{R}_{\hat{\boldsymbol{k}}}(\psi)^T\ \boldsymbol{R}_{\hat{\boldsymbol{k}}}(\psi)\ \boldsymbol{r}_2 = \\ &= \boldsymbol{r}_1^T\ \boldsymbol{I}\ \boldsymbol{r}_2 = \\ &= \boldsymbol{r}_1 \cdot \boldsymbol{r}_2\end{aligned}$$

Since rotation preserves vector magnitudes, we have $||\boldsymbol{r}_{1,rot}|| = ||\boldsymbol{r}_1||$ and $||\boldsymbol{r}_{2,rot}|| = ||\boldsymbol{r}_2||$. Therefore:

$$\cos\phi_{rot} = \frac{\boldsymbol{r}_{1,rot} \cdot \boldsymbol{r}_{2,rot}}{||\boldsymbol{r}_{1,rot}||\ ||\boldsymbol{r}_{2,rot}||} = \frac{\boldsymbol{r}_1 \cdot \boldsymbol{r}_2}{||\boldsymbol{r}_1||\ ||\boldsymbol{r}_2||} = \cos\phi$$

This shows that the angle, and the distance, between the two vectors remains unchanged after rotation: this procedure is valid for every point on the trajectory and on the target, thus confirming that the relative geometry of the trajectory is preserved.

Reference frames

The most important part of the optimization is the correct definition of the reference frame used to map the real trajectories in facility: a total of 5 reference frame are being used in the current implementation. For this reason, it is extremely important describing how the frames are created and why they are necessary.

$\{N\}$ – Inertial reference frame Starting from the inertial frame, it is used in the definition of the real trajectories, since these data are defined in a target centered inertial reference frame. The original problem is given in this frame, where the Sun direction is not static;

$\{S\}$ – Sun-fixed reference frame This frame is created in order to define a static direction of the Sun. Starting from the inertial reference frame, a time varying rotation matrix $\boldsymbol{R}_{S/N}$ can be defined from the variation of the Sun direction and it is used to rotate all the quantities in the new frame. The transport law is used to rotate velocities.

This frame is the last used in the pre-processing of the trajectories and the data can be exported for better ease of use inside the optimization, since its values do not change thorough the optimization;

$\{F\}$ – Facility reference frame This frame is static, if compared to **S**, and it is used to rotate and align the constant Sun direction with the facility's lamp direction. This is the only change of reference frame which happens after the pre-processing and before the optimization, it can be employment as soon as the lamp's information are known, so it is usually computed outside the optimization loops;

$\{R\}$ – Rotated Facility reference frame Entering now inside the optimization, the **R** frame uses the x_1 and x_3 solutions to rotate and translate the *cloud* along the Lamp direction. This frame is the most important, since it is the last required to obtain a complete characterization of the map. $\boldsymbol{R}_{R/F}$ is static;

$\{O\}$ – Omega reference frame last reference frame, was introduced after the first iterations, to expand the feasible solution domain. $\boldsymbol{R}_{O/R}$ is a rotating reference frame, applying the x_2 solution in the form of a angular velocity. This last reference frame allow to stretch the flown trajectory of the spacecraft inside the facility, allowing for a longer representation of highly vertical poses, transferring the high range of movements of GP12_1 onto GP12_2, or vice versa.

Other reference frame describes the spacecraft and the target body: $\{T\}$ for the target body and $\{C\}$ for the spacecraft (camera reference frame).

Some remarks regarding the reference frames are necessary: first of all, the choices regarding the number of reference frame is dictated by the MATLAB implementation, where keeping track of each frame is important. For this reason, it is clear how some frames could

be merged (e.g., $\{S\}$ and $\{F\}$ or $\{R\}$ and $\{O\}$), but a more readable implementation was chosen, leaving them as separate frames.

Lastly, it is important noting how both x_1 and x_2 act on the same behavior: the rotation around the Lamp direction. x_1 can be seen as the initial rotation angle, while x_2 corresponds to the evolution of it:

$$\gamma(t) = x_1 + x_2 \cdot t$$

This was also implemented on the *position* (x_3) of the cloud, leading to a linear velocity, which controls the time evolution of the position: it was chosen to remove it, since its implementation did not improve the results and the lighting conditions would have changed drastically during the testbed runs.

A quick summary of all the reference frames is reported in the following table:

Frame	Description	Definition	Type
$\{N\}$	Inertial	-	-
$\{S\}$	Sun-fixed	$\boldsymbol{R}_{S/N}(t)$	time dependant
$\{F\}$	Facility	$\boldsymbol{R}_{F/S}$	time independent
$\{R\}$	Rotated facility	$\boldsymbol{R}_{R/F}$	time independent
$\{O\}$	Omega	$\boldsymbol{R}_{O/R}(t)$	time dependent
$\{C\}$	Camera/Spacecraft	$\boldsymbol{R}_{N/C}(t)$	time dependant
$\{T\}$	Target	$\boldsymbol{R}_{N/T}(t)$	time dependant

Table 2.6: Frames specification.

2.3.2. Trajectory pre-processing

The first step of the optimization regards finding the trajectory to be optimized. The input of the optimization may consist of an entire mission: this data typically include correction maneuvers and similar discontinuities which have to be removed from the optimization data, since high torque robotic manipulators may introduce large errors when passing thorough non-continues paths. This is achieved by dividing the whole problem in smaller legs, via a slicer.

Analyzing now the complete problem, it is composed by:

- ${}^{N}\boldsymbol{y}_{sc}$ or ${}^{T}\boldsymbol{y}_{sc}$: spacecraft trajectory expressed both in inertial frame and in target body frame (both centered in the target body);

- $\boldsymbol{R}_{N/T}$: target body attitude;
- $\boldsymbol{t}$: time vector.

In this framework, it is necessary to compute the missing spacecraft attitude, creating a spacecraft pointing pose.

Slicing

Two main slicing conditions are found: maneuvering points and change of sign of latitude. La latter is imposed by the facility, since a change of hemisphere results in a different configuration for the mockup, which has to be reoriented on the end effector of the GP12 in order not to make the steel support of the end effector visible.

Starting from the maneuver split, its implementation regards finding the *specific angular momentum* discontinuities: $h = ||\boldsymbol{r} \times \boldsymbol{v}||$ is computed and the step is compared to a threshold.

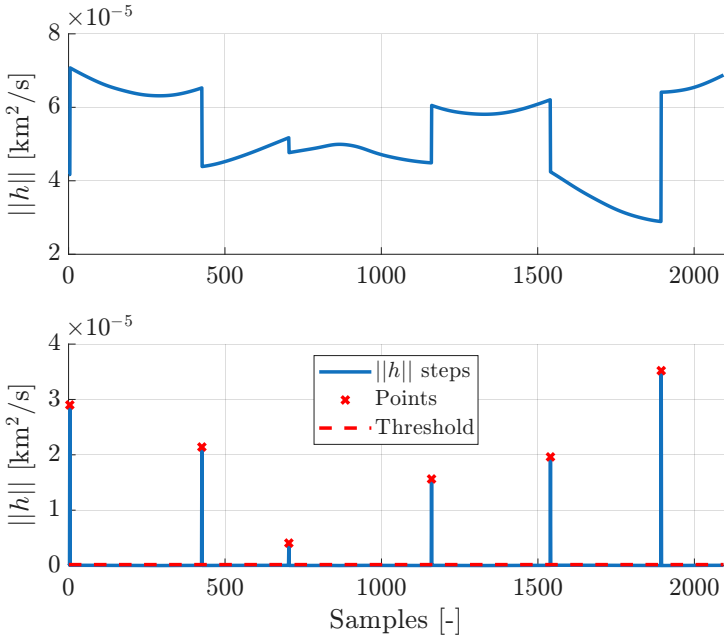


Figure 2.12: Specific angular momentum evolution.

Figure 2.12 shows the evolution of the magnitude of $\boldsymbol{h}$, for an example trajectory. It is important noting how this approach is not suited for plane change only maneuvers, since it relies only on the magnitude of $\boldsymbol{h}$. This, however, is not a problem, since these ideal maneuvers do not typically occur in the tested data.

However, a visual check is always important to ensure a correct maneuver detection.

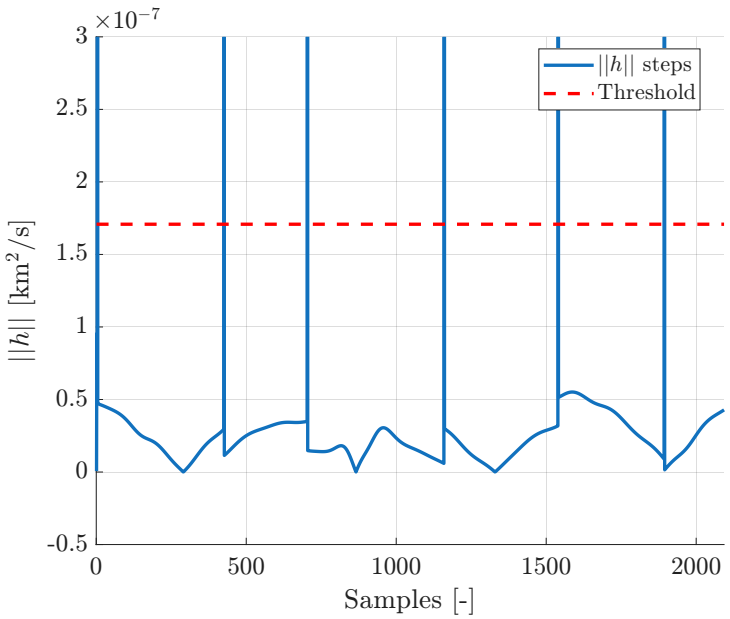


Figure 2.13: Specific angular momentum evolution (focus).

Figure 2.13 shows a focus of Figure 2.12, where the threshold is clearly visible and its value is sufficient to determine maneuvers and avoid false positives.

Following with the change of hemisphere, its implementation is trivial and consists in evaluating the change of sign of $\boldsymbol{r} \cdot \boldsymbol{s}$, where $\boldsymbol{s}$ is the major spin axis of the target body. This axis has been selected to coincide with the $J6$ axis of rotation of $GP12_2$, to take advantage of the extended range of rotation of the last joint for the optimization.

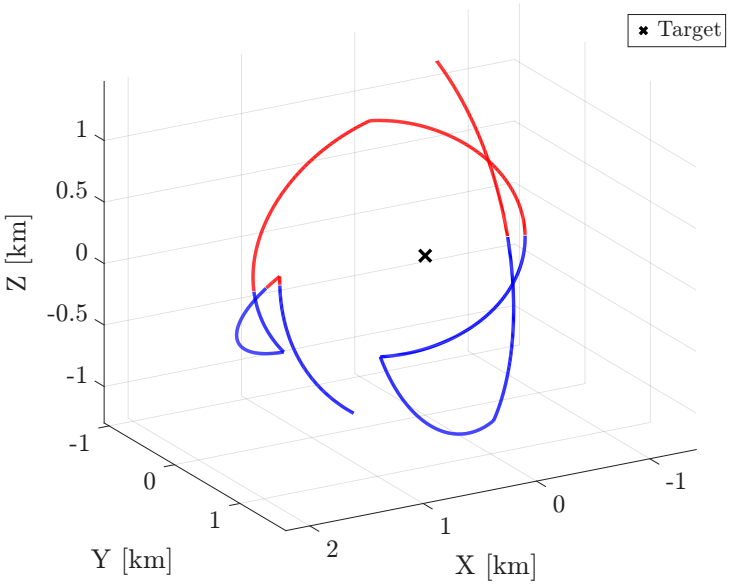


Figure 2.14: Hemisphere slicing.

The difference between the northern and the southern hemisphere trajectories can be appreciated in Figure 2.14, where red and blue lines are respectively reported. All the vectors used in this analysis have been defined in the target centered, inertial, reference frame.

The final outcome of the pre-processing can instead be seen in Figure 2.15, where each slicing condition has been applied to create individual legs.

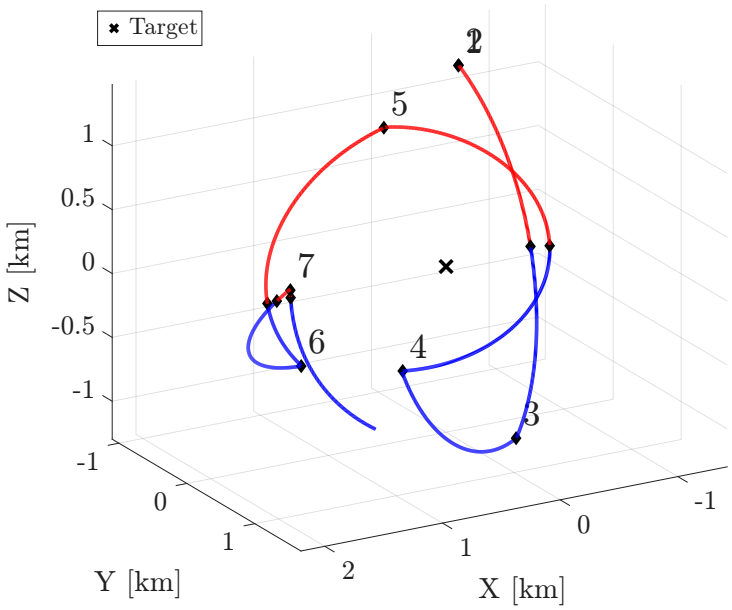


Figure 2.15: Sliced trajectory.

The numbered "diamond scatter" markers indicate the maneuver points, which define the start/end of each individual leg. Unlabeled diamonds mark hemisphere changes. A small overlapping between the first and second diamond is present, since the first maneuver occurs extremely close to the beginning of the whole trajectory.

An overlapping of the boundary points is also introduced, for a better facility representation: this can only be introduced in the change of hemisphere, since maneuvers may otherwise introduce discontinuities.

Spacecraft attitude

The spacecraft attitude is not available in the provided data and therefore needs to be assumed for the optimization: two different approaches are used to determine the rotation matrix $\boldsymbol{R}_{N/C}$, which relates the $\{C\}$ and the frame $\{N\}$.

While the $\hat{\boldsymbol{e}}_1$ component is always defined as the distance between the target and space-

craft position, normalized, and the $\hat{\boldsymbol{e}}_3$ completes the right hand triplet $\boldsymbol{e}_3 = \hat{\boldsymbol{e}}_1 \times \hat{\boldsymbol{e}}_2$, with $\hat{\boldsymbol{e}}_3 = \hat{\boldsymbol{e}}_3/||\hat{\boldsymbol{e}}_3||$, the choices for $\hat{\boldsymbol{e}}_2$ can vary.

A first approach uses the orbital plane to define $\hat{\boldsymbol{e}}_2$, using the difference between two consecutive ${}^N\boldsymbol{r}_C$ points to define the direction, which changes at each step. A second approach, instead, employs an unit vector parallel to the xy-plane of the $\{N\}$ frame to define $\hat{\boldsymbol{e}}_2$.

Both approaches are equally valid and can be used indiscriminately. For the implementation, the first approach has been selected.

Sun-fixed reference frame

Following what was introduced in section 2.3, the leg elements need to be defined in a Sun-fixed reference frame. This frame emulates the $\{F\}$ frame, where the lamp, acting as the Sun, is fixed in time.

Starting from the inertial reference frame of the legs, where every value is already defined, the change of reference frame has to be applied to the positions, velocities and attitude of both spacecraft and target body. The frame change is also applied to the Sun direction, in order to check for consistency.

Rotation matrix The rotation matrix $\boldsymbol{R}_{S/N}(t)$ has to be defined at each timestep, since the new reference frame is rotating. To accomplish this, the Equation 2.16 is employed in the matrix form: using the time changing Sun versor as $\hat{\boldsymbol{r}}_1(t)$ and a reference Sun versor as $\hat{\boldsymbol{r}}_2 = \hat{\boldsymbol{r}}_1(0)$, taken at the initial time. The cross product defines the instantaneous rotation angle $\boldsymbol{k}(t) = \hat{\boldsymbol{r}}_1(t) \times \hat{\boldsymbol{r}}_2$, with $\hat{\boldsymbol{k}}(t) = \boldsymbol{k}(t)/||\boldsymbol{k}(t)||$, and the rotation angle ψ is calculated with $\cos\psi(t) = \hat{\boldsymbol{r}}_1(t) \cdot \hat{\boldsymbol{r}}_2$ and $\sin\psi(t) = ||\hat{\boldsymbol{r}}_1(t) \times \hat{\boldsymbol{r}}_2||$.

However, the general Rodrigues formula is not well suited for small angles, where approximations may be introduced: using the second-order Taylor series expansion, the Rodrigues formula becomes:

$$\boldsymbol{R}(\psi) \approx \boldsymbol{I} + \psi[\hat{\boldsymbol{k}}]_\times + \frac{\psi^2}{2}[\hat{\boldsymbol{k}}]_\times^2$$

$$\boldsymbol{R}(\psi) \approx \boldsymbol{I} + [\boldsymbol{k}]_\times + \frac{1}{2}[\boldsymbol{k}]_\times^2, \quad \text{since} \quad [\boldsymbol{k}]_\times \approx \psi[\hat{\boldsymbol{k}}]_\times \tag{2.17}$$

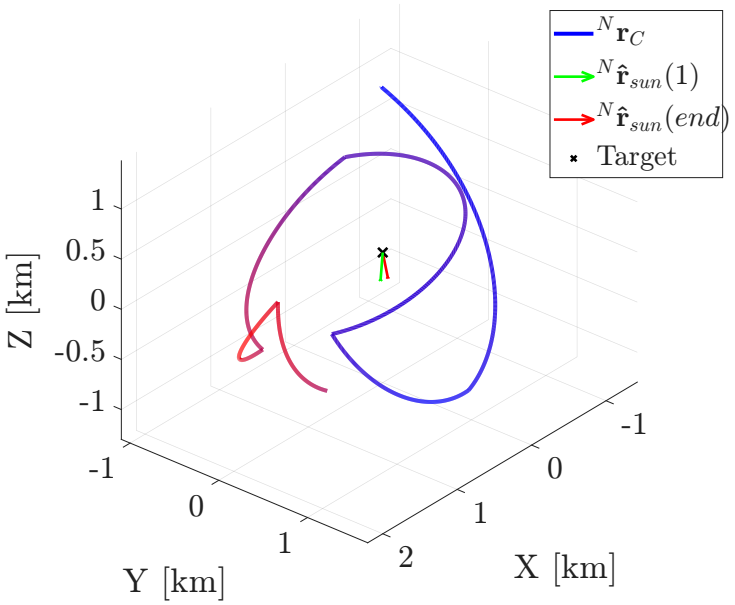


Figure 2.16: $\{N\}$ frame - ${}^N\boldsymbol{r}_C$ evolution with respect to the frame $\{S\}$.

Figure 2.16 shows how the vector ${}^N\boldsymbol{r}_C$ changes with respect to the rotating frame, indicating with a linear gradient (from blue to red) the normalized deviation. It is clearly shown the evolution between the first sun direction ${}^N\hat{\boldsymbol{r}}_{sun}(1)$ and the last ${}^N\hat{\boldsymbol{r}}_{sun}(end)$. The first value of the Sun versors have been chosen as the reference element and the following values are used to calculate the deviation to be compensated: this reference value is absolutely arbitrary and any other Sun versor could have been chosen, producing different trajectories in $\{S\}$, but equal in $\{F\}$.

In order to define the angular velocity vector of $\boldsymbol{R}_{S/N}(t)$, the following discrete approximation can be applied to the rotation matrix $\boldsymbol{R}(t)$, where each attitude matrix represents a discretized state of the system. Given two consecutive rotation matrices $\boldsymbol{R}(t_k)$, $\boldsymbol{R}(t_{k+1}) \in \mathrm{SO}(3)$ at times $t_k < t_{k+1}$, with time step $\Delta t = t_{k+1} - t_k$ [17]:

$$\boldsymbol{R}(t_{k+1}) = e^{[\hat{\boldsymbol{\omega}}]_\times \psi} \boldsymbol{R}(t_k) \tag{2.18}$$

and therefore

$$\boldsymbol{R}_{\text{rel}}(t_k) = \boldsymbol{R}(t_{k+1})\boldsymbol{R}(t_k)^T = e^{[\hat{\boldsymbol{\omega}}]_\times \psi} \tag{2.19}$$

The rotation angle is extracted as

$$\psi = \cos^{-1}\left(\frac{\mathrm{tr}(\boldsymbol{R}_{\mathrm{rel}}) - 1}{2}\right), \quad \psi \in [0, \pi] \tag{2.20}$$

The components of the angular velocity vector are then

$$\begin{aligned}
\boldsymbol{\omega}_x(t_k) &= \frac{\psi}{2\sin\psi \cdot \Delta t}\big(R_{32} - R_{23}\big) \\
\boldsymbol{\omega}_y(t_k) &= \frac{\psi}{2\sin\psi \cdot \Delta t}\big(R_{13} - R_{31}\big) \\
\boldsymbol{\omega}_z(t_k) &= \frac{\psi}{2\sin\psi \cdot \Delta t}\big(R_{21} - R_{12}\big)
\end{aligned} \tag{2.21}$$

where R_{ij} are the elements of $\boldsymbol{R}_{\mathrm{rel}}$.

This gives the constant angular velocity (in the spatial frame) that exactly maps $\boldsymbol{R}(t_k)$ to $\boldsymbol{R}(t_{k+1})$ over the interval Δt. The procedure can be extended to the more precise central finite difference scheme, which is used in the methodology. However, using the central finite difference scheme, the boundary points cannot be evaluated, since no boundary conditions are available. For this reason, the forward and backward finite difference schemes are used, respectively, for the first and the last nodes.

New legs Position and attitude vectors are rotated in the new rotating, Sun-fixed, reference frame $\{S\}$ by pre-multiplying the rotation matrix:

$$\begin{aligned}
{}^{S}\boldsymbol{r}_C &= \boldsymbol{R}_{S/N}{}^{N}\boldsymbol{r}_C \\
\boldsymbol{R}_{S/T} &= \boldsymbol{R}_{S/N}\boldsymbol{R}_{N/T} \\
\boldsymbol{R}_{S/C} &= \boldsymbol{R}_{S/N}\boldsymbol{R}_{N/C}
\end{aligned}$$

Concerning velocities, the **transport theorem** must be used, since the new reference frame is not static:

$${}^{S}\boldsymbol{v}_C = \boldsymbol{R}_{S/N}{}^{N}\boldsymbol{v}_C + \boldsymbol{\omega} \times {}^{N}\boldsymbol{r}_C \tag{2.22}$$

where $\boldsymbol{\omega} = \boldsymbol{\omega}(t)$ is the angular velocity vector of $\boldsymbol{R}_{S/N}$.

Scaling

In section 2.3.2, the real trajectories have been projected in the facility reference frame, which corresponds to the Sun-fixed reference frame. Now, it is important to scale the

physical quantities, in order to make the *cloud* fit inside the test facility.

Buckingham's Theorem can be employed for this result: first by identifying the physical quantities and then creating scaling parameters. The only quantities which requires a scaling are the position and velocity vectors of the spacecraft, and the time vector:

$$\begin{aligned}[\boldsymbol{r}_C] &= M^0 L^1 T^0 \\ [\boldsymbol{v}_C] &= M^0 L^1 T^{-1} \\ [\boldsymbol{t}] &= M^0 L^0 T^1\end{aligned}$$

Thus, a length and a time scaling need to be defined. Starting from the length scaling, this quantity is defined by the problem, since the real target body and the mockup target body are well defined.

$$\delta_L = \frac{L^*_{\text{mockup}}}{L^*_{\text{real target}}}$$

The other scaling parameter, δ_T, can be defined by identifying a desired *time of flight* of the facility simulation. By doing this, the time duration of a leg, after the application of the time scaling, will result in the predetermined total time. Following this analysis, all the dimensional quantities are scaled accordingly.

$$\begin{aligned}{}^N\boldsymbol{r}_C^{\text{scaled}}(t) &= {}^N\boldsymbol{r}_C(t) \cdot \delta_L \\ {}^N\boldsymbol{v}_C^{\text{scaled}}(t) &= {}^N\boldsymbol{v}_C(t) \cdot \delta_L/\delta_T \\ \boldsymbol{t}^{\text{scaled}} &= \boldsymbol{t} \cdot \delta_T\end{aligned}$$

This procedure is exactly the same, when performed for quantities expressed in any other frame reported in section 2.3.1.

2.3.3. Optimization

The facility optimization allows to dynamically replicate the real environment in the testbed simulation. This approach is based on pre-determined feasible trajectories, loaded as single legs, which are manipulated to create a robot-feasible path in the testbed. For this reason, the aim of the optimization is to enable the robot to follow the given trajectory, imposing collisions avoidance and feasibility constraints, instead of generating a facility feasible trajectory from scratch.

The testbed-mapped trajectory shall be compliant with the relative motion of the bodies.

Two constraints, which are defined by a mission simulator, cannot be altered in the facility optimization process: relative motion between the spacecraft and target body, and Sun direction with respect to the attitude of the target body.

Following what was discussed in section 2.3, the optimization parameters are the rotation angle x_1, the angular velocity x_2 and the translation x_3 of the *cloud* along the Sun direction: all collected in the $\boldsymbol{x}$ solution vector.

These variables represent the only degrees of freedom of the system and a change of these parameters does not alter the relative motion of the bodies, nor the Sun direction with respect to the target body, thus producing a physically identical configuration. For this reason, the optimization process can only act on two parameters: rotation and translation of the whole system along the Sun direction.

The general structure of the optimization is the following:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{s.t.} \quad \begin{cases} c(\boldsymbol{x}) \leq 0 \\ \boldsymbol{x}^{\text{LB}} \leq \boldsymbol{x} \leq \boldsymbol{x}^{\text{UB}} \end{cases} \tag{2.23}$$

where $f(\boldsymbol{x})$ is the cost function to be minimized, it evaluates the state $\boldsymbol{x}$ and produces a scalar value representing the quality of the facility configuration: in section 2.3.3 the cost function is described in detail.

This cost function is subject to nonlinear inequality constraints, which are described in section 2.3.3, and to bounds constraints, which limit the optimization variables within a predefined range: described in section 2.3.3.

Starting now with the schematic of the whole process, in the following figure, each component represents a key step.

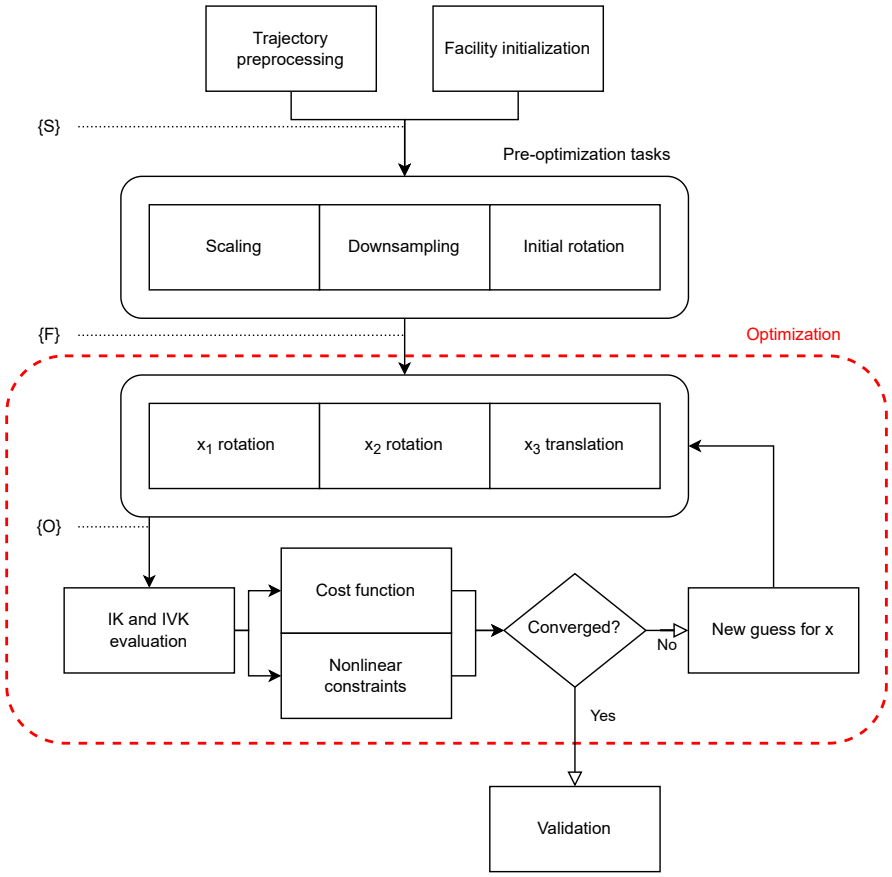


Figure 2.17: Methodology structure.

The main idea behind the optimization is connected to the reorientation of the trajectory cloud inside the testbed area. Starting from all the trajectory points and attitudes of the body, they are translated and rotated in space. This alteration of the cloud is a macroscopic effect only, and affects how the robotic manipulators move in the testbed, leaving internal dynamics of the leg unchanged.

This geometric approach allow to create an always feasible orbital mechanics behavior, while the optimization aims at finding the best orientation possible inside the facility, given feasibility constraints.

Lamp

The lamp details are not optimized inside the main optimization, since its values are common for all the single legs: it is placed manually in the facility. It simulates Sun illumination and it is described by two vectors: one regarding the position, and one

regarding the light direction.

Between position, Azimuth and Elevation of the light, a total of 5 additional parameters can be optimized. This strategy was tested for a single leg, producing excellent results, but its implementation for the complete problem shall be extended to an *external* approach: the outer optimization controls the lamp, while the inner optimization controls the trajectory mapping. This two-layer behavior is necessary, since the lamp details are common for all the individual legs, but it may introduce enormous computational costs. For this reason, it was chosen not to proceed with the final implementation, leaving the *lamp optimization* as a working separate function for a single leg approach.

Trajectory mapping

The *mapper-core* is the main function of the whole optimization: it uses the solution vector $\boldsymbol{x}$ to transform the original trajectory into the joint space solution, along calculating all the needed quantities for the cost function, the nonlinear constraints and the validation.

The input of the function is the *cloud* defined in the $\{F\}$ reference frame, while the rotation matrices $\boldsymbol{R}_{R/F}$ and $\boldsymbol{R}_{O/R}$ are calculated and then applied to the leg. While position vectors and attitudes are rotated by simply pre-multiplying the rotation matrices, the velocity vectors are rotated with the transport formula.

Once the frames are defined and the quantities have been rotated in the final $\{O\}$ frame, the task-space trajectory is calculated in the joint space: $\boldsymbol{\theta}^{\mathrm{GP12}_1}$ and $\boldsymbol{\theta}^{\mathrm{GP12}_2}$ are solved with the IK, while $\dot{\boldsymbol{\theta}}^{\mathrm{GP12}_1}$ and $\dot{\boldsymbol{\theta}}^{\mathrm{GP12}_2}$ are found with the IVK.

This function is also used to evaluate additional quantities, by requesting two outputs: this way, the complete set of positions and attitudes of the end effectors can be retrieved.

Cost function

The cost function of the optimization is extremely important since drives the convergence of the solution. Two main characteristics have been identified in order to obtain a good parameter of merit: minimization of singularities and smoothness.

Technically speaking, the singularities of the robots cannot be avoided if the end effector's poses are already defined, since the analytical IK allows for a single solution. This is why the optimization shall reorient the *cloud* in such a way the robots are given singularity-free trajectories. This can be achieved by using one of the manipulability parameters discussed in section 2.1.5.

The μ_1 parameter is perfectly suited for this purpose, since its value gets closer to 1 whenever the robot is further away from mono-directional singularities (indicating an isotropic manipulability ellipsoid), while it grows to infinity if a singularity is reached.

However, μ_1 may fail to identify certain symmetric singularities (where the ellipsoid collapses uniformly across all directions), whereas μ_3 would produce a more reliable and global measure of the overall kinematic capability. Fortunately, this symmetric singularities, are basically impossible to occur, and μ_1 remains the best choice between the two.

Nonlinear constraints

The output of the core-mapper allow to evaluate the nonlinear constraints of the optimization problem: it is important reminding how all these values are inequality constraints, related to the robotic domain of the problem. This behavior is expected, since the natural constraints, related to astrodynamics consistency, are not being checked, since their characteristics are unaltered by the optimization and will be verified with the validation.

The constraints are driven by the GP12 limits: in term of maximum and minimum deflection angles of the joints, as well as in terms of collision avoidance with other facility elements or with the facility walls.

The following equations summarize the implemented constraints. The total number adds up to 161 scalar equations, which are being computed and tested for each time instant $0 \leq k \leq N$ of the given leg to be optimized. Superscripts or explicit variable dependence are not used, for a better readability.

$$
\boldsymbol{c}_k^{(161\times1)}(\boldsymbol{x}) = \begin{cases}
\boldsymbol{z}_1^{(6\times1)} := \boldsymbol{\theta}_1 - \boldsymbol{\theta}_{lim,1}^{+} + \boldsymbol{\delta}_\theta \leq \mathbf{0} \\
\boldsymbol{z}_2^{(6\times1)} := -\boldsymbol{\theta}_1 + \boldsymbol{\theta}_{lim,1}^{-} + \boldsymbol{\delta}_\theta \leq \mathbf{0} \\
\boldsymbol{z}_3^{(6\times1)} := \boldsymbol{\theta}_2 - \boldsymbol{\theta}_{lim,2}^{+} + \boldsymbol{\delta}_\theta \leq \mathbf{0} \\
\boldsymbol{z}_4^{(6\times1)} := -\boldsymbol{\theta}_2 + \boldsymbol{\theta}_{lim,2}^{-} + \boldsymbol{\delta}_\theta \leq \mathbf{0} \\
\boldsymbol{z}_5^{(6\times1)} := |\dot{\boldsymbol{\theta}}_1| - \dot{\boldsymbol{\theta}}_{lim,1} + \boldsymbol{\delta}_{\dot{\theta}} \leq \mathbf{0} \\
\boldsymbol{z}_6^{(6\times1)} := |\dot{\boldsymbol{\theta}}_2| - \dot{\boldsymbol{\theta}}_{lim,2} + \boldsymbol{\delta}_{\dot{\theta}} \leq \mathbf{0} \\
\boldsymbol{z}_7^{(1\times1)} := d_{EE_1} - d_{EE_1}^{max} + \boldsymbol{\delta}_{\text{distance}} \leq 0 \\
\boldsymbol{z}_8^{(1\times1)} := d_{EE_2} - d_{EE_2}^{max} + \boldsymbol{\delta}_{\text{distance}} \leq 0 \\
\boldsymbol{z}_9^{(1\times1)} := -\boldsymbol{p}_{EE_1} \cdot [0\ 0\ 1] + \boldsymbol{\delta}_{\text{distance}} \leq 0 \\
\boldsymbol{z}_{10}^{(1\times1)} := -\boldsymbol{p}_{EE_2} \cdot [0\ 0\ 1] + \boldsymbol{\delta}_{\text{distance}} \leq 0 \\
\boldsymbol{z}_{11}^{(1\times1)} := -\boldsymbol{p}_{J_{5,1}} \cdot [0\ 0\ 1] + \boldsymbol{\delta}_{\text{distance}} \leq 0 \\
\boldsymbol{z}_{12}^{(1\times1)} := -\boldsymbol{p}_{J_{5,2}} \cdot [0\ 0\ 1] + \boldsymbol{\delta}_{\text{distance}} \leq 0 \\
\boldsymbol{z}_{13}^{(49\times1)} := -||\boldsymbol{p}_{J_{i,1}} - \boldsymbol{p}_{J_{j,2}}|| + \boldsymbol{\delta}_{\text{robot}} \leq 0, \quad \forall (i,j) \in \mathcal{I}_1 \times \mathcal{I}_2 \\
\boldsymbol{z}_{14}^{(7\times1)} := -\boldsymbol{\phi}_{J_{i,1}}^{\text{lamp}} + \boldsymbol{\delta}_{\text{phase}} \leq 0 \\
\boldsymbol{z}_{15}^{(7\times1)} := -||\boldsymbol{p}_{J_{i,1}} - \boldsymbol{p}_{\text{lamp}}|| + \boldsymbol{\delta}_{\text{distance}} \leq 0, \quad i \in \mathcal{I}_1 \\
\boldsymbol{z}_{16}^{(7\times1)} := -\boldsymbol{n}_{\text{wall}_1} \cdot \left(\boldsymbol{p}_{J_{i,1}} - \boldsymbol{p}_{\text{wall}_1}\right) + \boldsymbol{\delta}_{\text{distance}} \leq 0, \quad i \in \mathcal{I}_1 \\
\boldsymbol{z}_{17}^{(7\times1)} := -\boldsymbol{n}_{\text{wall}_2} \cdot \left(\boldsymbol{p}_{J_{i,1}} - \boldsymbol{p}_{\text{wall}_2}\right) + \boldsymbol{\delta}_{\text{distance}} \leq 0, \quad i \in \mathcal{I}_1 \\
\boldsymbol{z}_{18}^{(7\times1)} := -\boldsymbol{n}_{\text{wall}_3} \cdot \left(\boldsymbol{p}_{J_{i,1}} - \boldsymbol{p}_{\text{wall}_3}\right) + \boldsymbol{\delta}_{\text{distance}} \leq 0, \quad i \in \mathcal{I}_1 \\
\boldsymbol{z}_{19}^{(7\times1)} := -\boldsymbol{n}_{\text{wall}_4} \cdot \left(\boldsymbol{p}_{J_{i,1}} - \boldsymbol{p}_{\text{wall}_4}\right) + \boldsymbol{\delta}_{\text{distance}} \leq 0, \quad i \in \mathcal{I}_1 \\
\boldsymbol{z}_{20}^{(7\times1)} := -\boldsymbol{n}_{\text{wall}_1} \cdot \left(\boldsymbol{p}_{J_{j,2}} - \boldsymbol{p}_{\text{wall}_1}\right) + \boldsymbol{\delta}_{\text{distance}} \leq 0, \quad j \in \mathcal{I}_2 \\
\boldsymbol{z}_{21}^{(7\times1)} := -\boldsymbol{n}_{\text{wall}_2} \cdot \left(\boldsymbol{p}_{J_{j,2}} - \boldsymbol{p}_{\text{wall}_2}\right) + \boldsymbol{\delta}_{\text{distance}} \leq 0, \quad j \in \mathcal{I}_2 \\
\boldsymbol{z}_{22}^{(7\times1)} := -\boldsymbol{n}_{\text{wall}_3} \cdot \left(\boldsymbol{p}_{J_{j,2}} - \boldsymbol{p}_{\text{wall}_3}\right) + \boldsymbol{\delta}_{\text{distance}} \leq 0, \quad j \in \mathcal{I}_2 \\
\boldsymbol{z}_{23}^{(7\times1)} := -\boldsymbol{n}_{\text{wall}_4} \cdot \left(\boldsymbol{p}_{J_{j,2}} - \boldsymbol{p}_{\text{wall}_4}\right) + \boldsymbol{\delta}_{\text{distance}} \leq 0, \quad j \in \mathcal{I}_2
\end{cases} \tag{2.24}
$$

It can be seen how each element has an additional quantity, denoted by the Greek letter δ. These vectors are composed of statically defined values, applied to each element of the vectorized constraint, which introduce a safety margin. These values introduce an additional layer of safety against constraint violations: indeed, *fmincon* usually violates the nonlinear constraints by a small gap (always smaller than 1×10^{-10}) and this margin is needed to ensure complete feasibility for any converged solution that slightly violates the constraints.

Equation 2.24 shows the complete set of nonlinear constraints evaluated at each trajectory point k. Each family of nonlinear inequality constraints is analyzed in details in the following paragraphs.

Angle limits The first 24 constraints ($\boldsymbol{z}_1$ to $\boldsymbol{z}_4$) regards joint angles limits, while the following 12 constraints ($\boldsymbol{z}_5$ and $\boldsymbol{z}_6$) regards joint angular velocity limits. The $\boldsymbol{\theta}$ values of both robots are computed with the $\boldsymbol{x}$ solution and their values are compared against the facility limits. The margins for $\boldsymbol{\delta}_\theta$ and $\boldsymbol{\delta}_{\dot{\theta}}$ are, respectively, 5° and 1 rad s.

Distance margin Constraints $\boldsymbol{z}_7$ and $\boldsymbol{z}_8$ regards the reachability condition, evaluating the distance margin between desired and actual end-effector position: less or equal than zero, if the target is reachable, greater than zero, otherwise. The distance margin $\boldsymbol{\delta}_{\text{distance}}$ of 0.2 m is employed.

Floor collision avoidance Regarding the collision avoidance with the floor, constraints $\boldsymbol{z}_9$ to $\boldsymbol{z}_{11}$ check the positions of the end effectors and the positions of the fifth joints with respect to the $z = 0$ facility plane. The safety margin $\boldsymbol{\delta}_{\text{distance}}$ is being used.

Robot collision avoidance $\boldsymbol{z}_{13}$ describes the collision avoidance constraints between the joints of GP12_1 and GP12_2. These values are the unique distances between the joints, which are a total of 49, given the total number of analyzed positions, for each robot, to be 7. It is important noting how the collision avoidance of the robot with its own structure is not important and redundant, since the joint angle limits inherit this constraints and do not allow for self collision. Here, $\boldsymbol{\delta}_{\text{robot}}$ of 0.3 m is used.

Lamp phase constraints Illumination of the target is extremely important and, for this reason, the GP12_1 must not interfere with the light beam: the angle between the joints of GP12_1 and the lamp, with respect to the target body, is calculated and its value $\boldsymbol{\phi}^{lamp}_{J_i,1}$ is checked to be higher than $\boldsymbol{\delta}_{\text{phase}}$. This *non shadow condition* ensures the first robot is not interfering with the lighting: allowing to discard conditions where the camera in casting its own shadow on the target or conditions where the camera is obstructed by the lamp. The margin $\boldsymbol{\delta}_{\text{phase}}$ consists of 20°.

Lamp collision avoidance Last constraint regarding the lamp is the collision avoidance: $\boldsymbol{z}_{15}$ checks the distance of the GP12_1 joints with respect to the lamp, and these values are compared with a safety margin $\boldsymbol{\delta}_{\text{distance}}$. Only GP12_1 is being checked, since the minimum distance discussed in section 2.3.3 is sufficient for the GP12_2 collision avoidance with the lamp.

Walls collision avoidance The wall collision avoidance is extremely important, since the facility is located inside a protective cage and collision with it is technically possible.

$\boldsymbol{z}_{16}$ to $\boldsymbol{z}_{23}$ define these conditions, analyzing the combinations of 4 walls and a total of 14 robot positions across the two robots. For this last constrain, $\boldsymbol{\delta}_{\text{distance}}$ is used, while the walls locations can be found in Table 2.5.

Table 2.7 summarizes all the $\boldsymbol{\delta}_x$ parameters used in the nonlinear constraints:

Description	Symbol	Value
θ margin	$\boldsymbol{\delta}_\theta$	$5°$
$\dot{\theta}$ margin	$\boldsymbol{\delta}_{\dot{\theta}}$	$1\,\text{rad}\,\text{s}^{-1}$
Distance margin	$\boldsymbol{\delta}_{\text{distance}}$	$0.2\,\text{m}$
Robot distance margin	$\boldsymbol{\delta}_{\text{robot}}$	$0.3\,\text{m}$
Phase margin	$\boldsymbol{\delta}_{\text{phase}}$	$20°$

Table 2.7: Nonlinear constraints margins.

Constraints are imposed on all trajectory points to guarantee feasibility for the robots throughout the entire path. However, they do not modify the trajectory geometry, which remains fixed in terms of relative positions and phase angles and is not subject to change in the optimization. This is crucial and extremely important, since the goal of the facility optimization is to adapt the environment to the given trajectory, not to change the trajectory itself.

Bounds constraints

The bounds constraints are needed for the optimizer to identify a limit for the optimization variables. While x_1 is confined between 0 and 2π for obvious reasons, $-0.1 \leq x_2 \leq 0.1$ since higher values are simply not of interest.

Finally x_3 is defined between 0.2 and 1, since it describes the position of the target body with respect to the lamp, between a predefined minimum and maximum distance. This value is 0 if the target body is directly located inside the lamp, while is 1 if it is located at the maximum distance. For this quantity, it was chosen to define a minimum of 0.2, since closer configurations would result in unfeasible solutions.

Initial guess

The initial guess is needed for the *fmincon* function to start the optimization, and its values can all be initialized to zero, since the optimization proved to be robust enough to converge for any given initial values. However, $\boldsymbol{x}_0$ can also be chosen wisely: starting from $x_{0,2}$, it can be initialized to 0, since x_2 is expected to be small and its sign is not

fixed. Meanwhile, x_3 is expected to be close to 0.5, since this value corresponds to a mean distance from the lamp, indicating a symmetric positioning with respect to both robots: $x_{0,3} = 0.5$.

Concerning x_1, instead, a small minimization can be performed, in order to align the spacecraft trajectory with GP12_1. The process is simple and consists in minimizing the inverse of the dot product between the $\text{GP12}_2 \rightarrow \text{GP12}_1$ direction and ${}^R\hat{\boldsymbol{r}}_C(t_{N/2})$ direction, which depends on x_1. ${}^R\hat{\boldsymbol{r}}_C(t_{N/2})$ corresponds to the normalized position vector of the spacecraft, evaluated in the $\{R\}$ frame and in the middle time interval. This procedure is sufficient to define a good guess of $x_{0,1}$.

Complete optimization and optimization slicer

Although single legs are assumed to be quite small in terms of distance of travel for the end effectors of the robots, the convergence of each one is not guaranteed. For this reason, an additional modification shall be introduced.

It was decided to add a leg slicer, whenever the optimization would not converge. The working principle is trivial and consists in splitting the non converged leg into two sub-legs: once this is computed, the optimizer analyzes both new legs. This procedure is performed every time the optimizer reaches the maximum function evaluation or maximum iteration count. A maximum split depth of 3 is selected as the maximum number of times a leg can be sliced. This number ensures the sliced leg, if not feasible at all, to be skipped. For reference, a 3 times sliced leg would be $\frac{1}{2^3}$ of the original length.

Moreover, the slicer takes into account an overlapping between the newly sliced trajectories: a defined number of overlapping points is added to the beginning and/or to the end of the new data. This way, testbed implementations can take advantage of the full problem, without losing data. This procedure proved to be robust, leading to a full optimization of the entire given trajectory.

Downsampling

The optimization is usually run with the untouched and complete set of leg points: this is the safest procedure, but also the slowest. The optimization, in fact, proved to be able to determine a similar result by using a downsampled leg. This procedure consists in selecting a smaller number of leg points, without interpolating, and maintaining the boundary points. $\boldsymbol{x}$ solution is expected to be similar to the original problem, since feasibility and cost functions are evaluated on robotic mechanics and continuity is not important. The only, and important, problem regards singularities: by downsampling the

trajectory, singularities may not be correctly represented and the optimization may fail to evaluate them. Fortunately, this behavior can be reduced by using a minium number of downsampled points: setting it to 50, as an educated guess, proved to be a good number.

Downsampling is mainly used to speed up convergence, with computation time decreasing roughly in proportion to the number of points (as observed in numerical simulations). Finally, the $\boldsymbol{x}$ solution is applied to the whole leg, evaluating the nonlinear constraints and validating the solution. This is also important for validation purposes, since the downsampled trajectory may be interpolated to generate back the optimized discarded points, leaving with the complete set to be compared for error evaluation.

2.3.4. Validation

The solution of the facility optimization is employed to reconstruct the complete, original problem. By doing so, it can be compared with the input trajectories to check for errors. If the outcome of V&V is compliant with small errors, then the solutions can be used in the RAFFAELLO facility to produce datasets.

There are two validations to be performed: robotics validation and trajectory validation; carried out in this order to first validate the kinematics of the digital twin and then the methodology as a whole. Trajectory validation is self explanatory and consists in reconstructing trajectory of the spacecraft, attitude of the spacecraft and Sun direction both in $\{T\}$ frame and in $\{N\}$ frame, in order to compare them with the original problem.

Camera sensor emulation, instead, consists in creating a mathematical model of the camera sensor, using projection formulas, in order to emulate the FOV of the real camera and test it with a LED dot pattern target, which can be mounted on the $GP12_2$ end effector. This step serves as an additional check to verify that the overall reproduction behaves correctly in the facility and to test pointing accuracy.

Robotics validation

Starting with the *Robotics validation*, the results of the IVK are compared to the numerical differentiation. This comparison is important to verify the correctness of the $\dot{\boldsymbol{\theta}}$ joint space result, since these values are used in the nonlinear constraints evaluation, which undergo Yaskawa safety limits. This step of the validation is also important to verify the correctness of the Jacobian, since it is crucial for a correct cost function evaluation.

The procedure consists in generating a custom trajectory from scratch, consisting in positions of the end effector: this path is then differentiated, to obtain the velocities. The

very same procedure is carried out for the euler angles, corresponding to the attitude and the twist of the end effector. The problem is now completely defined in the task domain, similarly to the mapping problem. The task space solution is mapped to the joint space solution from the two different sides: with IK and with IVK to obtain the joint space solution relative to $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$, respectively. In order to validate the map, the $\dot{\boldsymbol{\theta}}$ solution can be reconstructed via differentiation from $\boldsymbol{\theta}$, or, vice versa, $\boldsymbol{\theta}$ can be retrieved from $\dot{\boldsymbol{\theta}}$ by integration.

This last step is the validation, proving the Jacobian to be correct. The scheme of the approach can be found in Figure 2.18.

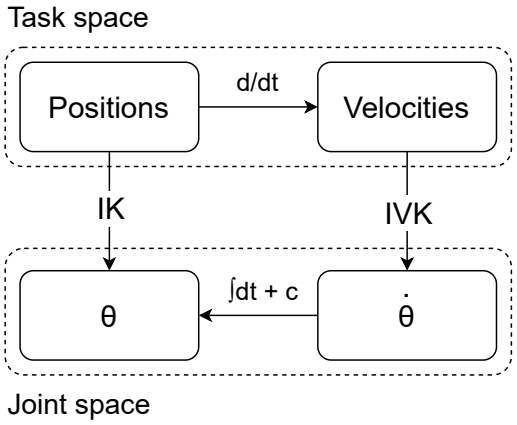


Figure 2.18: $\boldsymbol{J}$ validation scheme.

Now that the procedure is clear, the results are shown in the following plots.

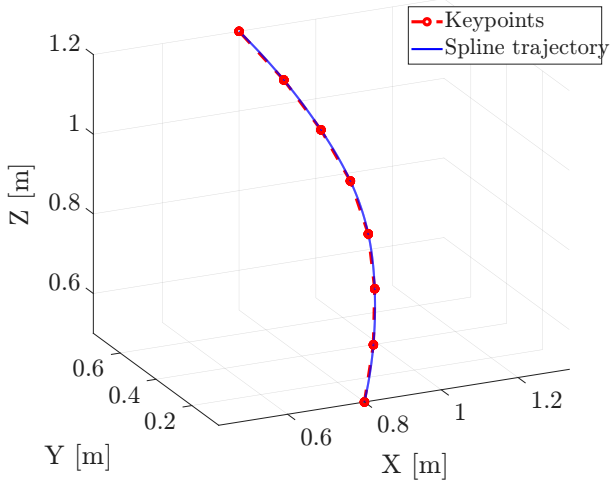


Figure 2.19: $\boldsymbol{J}$ validation: custom trajectory.

8 points are defined and the smooth trajectory is then calculated using tridimensional splines, in order to adjust the total number of points for sensitivity analysis.

The IK and FK are then evaluated, to validate the reconstructed positions:

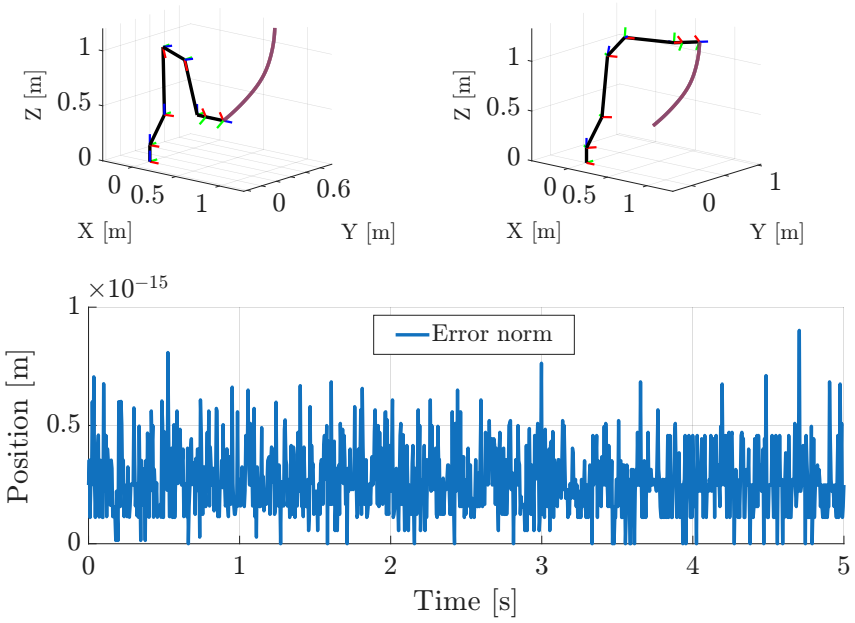


Figure 2.20: FK reconstruction and validation.

Finally, the IK and IVK are tested with the aformentioned procedure, to verify the correctness of the Jacobian function.

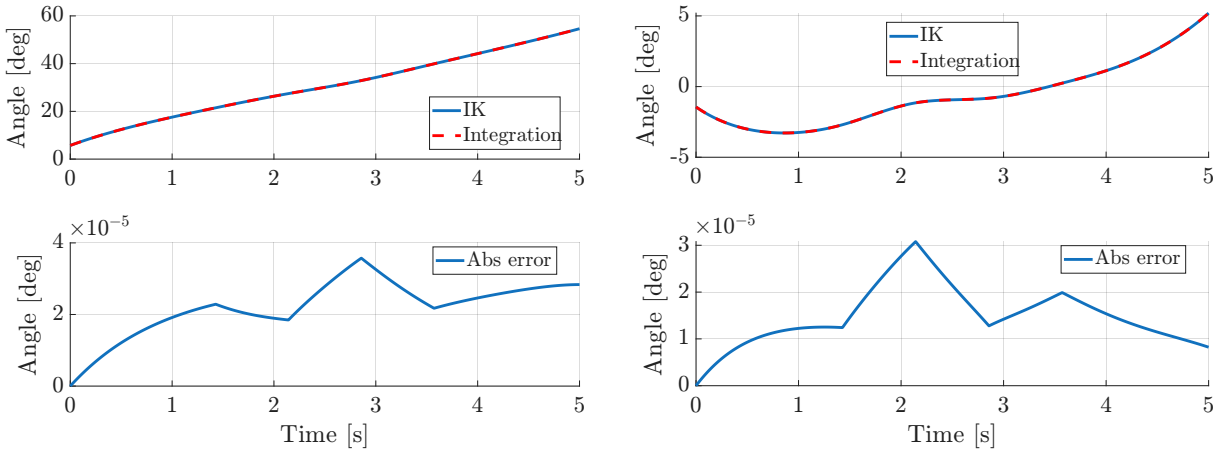


Figure 2.21: $\boldsymbol{J}$ validation: joint 1 and 2.

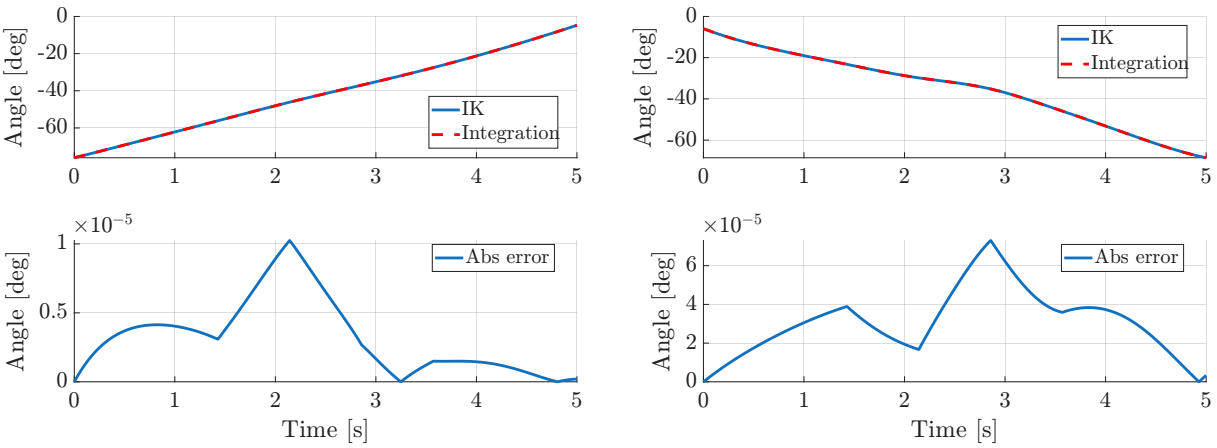


Figure 2.22: $\boldsymbol{J}$ validation: joint 3 and 4.

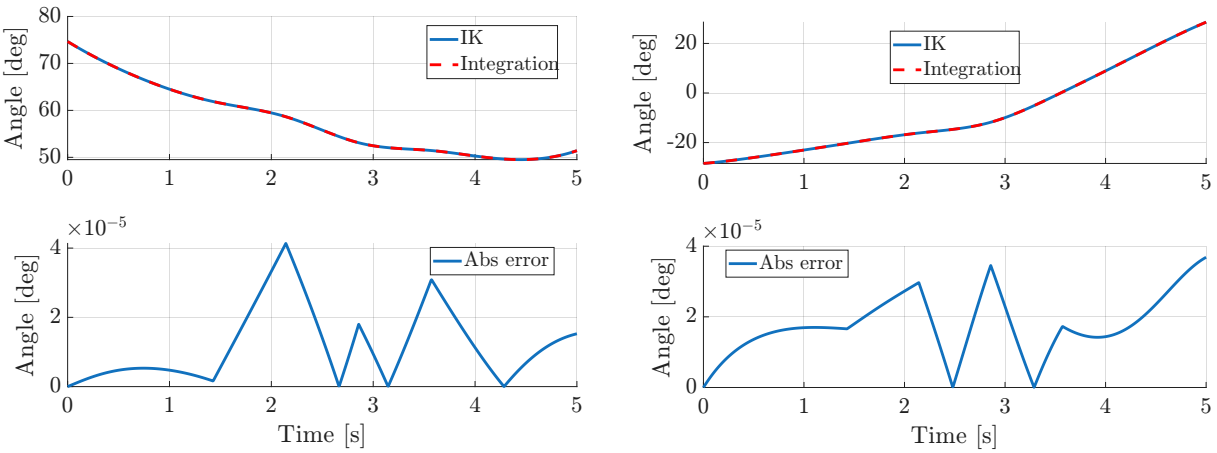


Figure 2.23: $\boldsymbol{J}$ validation: joint 5 and 6.

It can be seen how the errors, for each Joint solution, are small and no offset is visible, thus validating the functions.

Trajectory validation

The $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ joint solutions are used in the FK to generate the positions and attitude of both spacecraft (camera) and target body (mockup). The first robot, GP12_1, generates $\boldsymbol{R}_{O/C}(t)$ and ${}^{O}\boldsymbol{y}_C(t)$ while, GP12_2, generates $\boldsymbol{R}_{O/T}(t)$ and ${}^{O}\boldsymbol{y}_T(t)$. These are the poses of the payloads of both end effectors, consisting in a camera and a mockup of the target body. The direction of the lamp ${}^{O}\hat{\boldsymbol{r}}_{\text{sun}}(t)$, instead, is known and fixed in this reference frame.

It can be immediately seen how the matrix $\boldsymbol{R}_{O/T}(t)$ can be used to rotate all the quantities in $\{T\}$, allowing for a quick comparison with the trajectories given in this frame.

This procedure is trivial and consists in pre-multiplying ${}^{O}\boldsymbol{r}_C(t)$, ${}^{O}\hat{\boldsymbol{r}}_{\text{sun}}(t)$ and $\boldsymbol{R}_{O/C}(t)$ by $\boldsymbol{R}'_{O/T}(t)$:

$$\begin{aligned} {}^{T}\boldsymbol{r}_C(t) &= \boldsymbol{R}'_{O/T}(t)\left({}^{O}\boldsymbol{r}_C(t) - {}^{O}\boldsymbol{r}_T(t)\right) \\ \boldsymbol{R}_{T/C}(t) &= \boldsymbol{R}'_{O/T}(t)\boldsymbol{R}_{O/C}(t) \\ {}^{T}\hat{\boldsymbol{r}}_{\text{sun}}(t) &= \boldsymbol{R}'_{O/T}(t){}^{O}\hat{\boldsymbol{r}}_{\text{sun}}(t) \end{aligned}$$

Concerning ${}^{T}\boldsymbol{v}_C(t)$, it is calculated with the transport formula (explained in Equation 2.22), using $\boldsymbol{\omega}_{T/O}(t)$ (explained in Equation 2.21):

$${}^{T}\boldsymbol{v}_C(t) = \boldsymbol{R}'_{O/T}(t){}^{O}\boldsymbol{v}_C(t) + \boldsymbol{\omega}_{T/O}(t) \times {}^{T}\boldsymbol{r}_C(t)$$

With $\boldsymbol{y}_C(t) = [\boldsymbol{r}_C(t)\ \boldsymbol{v}_C(t)]$, being the complete state of the spacecraft, composed by positions and velocities: this is a $N \times 6$ vector.

It is important noting how these quantities are defined in the facility scaled environment and the real elements are reconstructed with:

$$\begin{aligned} {}^{T}\boldsymbol{r}_C^{\text{real}}(t) &= {}^{T}\boldsymbol{r}_C(t) \cdot (\delta_L)^{-1} \\ {}^{T}\boldsymbol{v}_C^{\text{real}}(t) &= {}^{T}\boldsymbol{v}_C(t) \cdot (\delta_L/\delta_T)^{-1} \\ \boldsymbol{t}^{\text{real}} &= \boldsymbol{t} \cdot (\delta_T)^{-1} \\ \boldsymbol{R}_{T/C}^{\text{real}}(t) &= \boldsymbol{R}_{T/C}(t) \\ {}^{T}\hat{\boldsymbol{r}}_{\text{sun}}^{\text{real}}(t) &= {}^{T}\hat{\boldsymbol{r}}_{\text{sun}}(t) \end{aligned}$$

It can be seen how, the joint space solution is sufficient to reproduce the $\{T\}$ trajectory: this, however, is not valid for the reconstruction of the trajectory in $\{N\}$, where rotation matrices need to be calculated outside the post-processing environment.

Regarding the $\{N\}$ reconstruction, the baseline starts from the same steps performed for the $\{T\}$: calculating the poses of both robots to gather $\boldsymbol{R}_{O/C}(t)$, ${}^{O}\boldsymbol{y}_C(t)$, $\boldsymbol{R}_{O/T}(t)$ and ${}^{O}\boldsymbol{y}_T(t)$. The Sun vector is also known to be ${}^{O}\hat{\boldsymbol{r}}_{\text{sun}}(t)$. It is now necessary to define $\boldsymbol{R}_{N/O}(t)$.

$$\boldsymbol{R}_{N/O}(t) = \boldsymbol{R}_{N/S}(t)\boldsymbol{R}_{S/F}\boldsymbol{R}_{F/R}\boldsymbol{R}_{R/O}(t)$$

The first two rotation matrices are defined by the pre-processing and fed directly into the validation function, since it is not possible to reproduce them from the joint space

solution.

Remembering $\boldsymbol{R}_{F/R} = \boldsymbol{R}_{R/F}^T$, the Rodrigues matrix formulation (Equation 2.16) can be used to define $\boldsymbol{R}_{R/F}$, using x_1 as the rotation angle and the static Sun direction defined in the frame ${}^O\hat{\boldsymbol{r}}_{\text{sun}} = {}^R\hat{\boldsymbol{r}}_{\text{sun}} = {}^F\hat{\boldsymbol{r}}_{\text{sun}}$ as the rotation versor.

$\boldsymbol{R}_{R/O}(t)$ can be derived from $\boldsymbol{R}_{O/R}(t)$, since $\boldsymbol{R}_{R/O}(t) = \boldsymbol{R}_{O/R}^T(t)$. The Rodrigues matrix formulation is used, since the rotation axis is the static Sun direction and the angle is determined by the integration in time of x_2.

It can be noted how the rotation matrix $\boldsymbol{R}_{F/O}(t)$ ($\boldsymbol{R}_{O/F}(t)$, with $\boldsymbol{R}_{F/O}(t) = \boldsymbol{R}_{O/F}^T(t)$) could be retrieved via the Rodrigues matrix formulation, using $\psi(t) = x_1 + x_2 \cdot t$. However, for ease of implementation, it was decided not to proceed with this implementation. This was already discussed in the section 2.3.1.

Finally, the rotation can be applied using the same formula used for the $\{T\}$ analysis, now using $\boldsymbol{R}_{N/O}(t)$:

$$
\begin{aligned}
{}^N\boldsymbol{r}_C(t) &= \boldsymbol{R}_{N/O}(t)\left({}^O\boldsymbol{r}_C(t) - {}^O\boldsymbol{r}_T(t)\right) \\
\boldsymbol{R}_{N/C}(t) &= \boldsymbol{R}_{N/O}(t)\boldsymbol{R}_{O/C}(t) \\
{}^N\hat{\boldsymbol{r}}_{\text{sun}}(t) &= \boldsymbol{R}_{N/O}(t){}^O\hat{\boldsymbol{r}}_{\text{sun}}(t)
\end{aligned}
$$

And, for the velocities:

$$
{}^N\boldsymbol{v}_C(t) = \boldsymbol{R}_{N/O}(t){}^O\boldsymbol{v}_C(t) + \boldsymbol{\omega}_{N/O}(t) \times {}^N\boldsymbol{r}_C(t)
$$

Lastly, scaling is applied to produce real trajectories:

$$
\begin{aligned}
{}^N\boldsymbol{r}_C^{\text{real}}(t) &= {}^N\boldsymbol{r}_C(t) \cdot (\delta_L)^{-1} \\
{}^N\boldsymbol{v}_C^{\text{real}}(t) &= {}^N\boldsymbol{v}_C(t) \cdot (\delta_L/\delta_T)^{-1} \\
\boldsymbol{t}^{\text{real}} &= \boldsymbol{t} \cdot (\delta_T)^{-1} \\
\boldsymbol{R}_{N/C}^{\text{real}}(t) &= \boldsymbol{R}_{N/C}(t) \\
{}^N\hat{\boldsymbol{r}}_{\text{sun}}^{\text{real}}(t) &= {}^N\hat{\boldsymbol{r}}_{\text{sun}}(t)
\end{aligned}
$$

Camera sensor emulation

Final section of the validation: the camera sensor emulation. It involves defining some points in the $\{T\}$ frame and evaluating how they are seen by the camera sensor, in the $\{C\}$ frame.

The map $\mathbb{R}^3 \mapsto \mathbb{R}^2$, created when the $[X\ Y\ Z]$ points are projected to the camera sensor coordinates $[u\ v]$, involves a reduction of space dimensions, deriving:

$$\begin{cases} x = X/Z \\ y = Y/Z \end{cases}$$

By doing so, the standard equation for the sensor modeling [13], without any distortion, is:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{cases} f_x\ x + c_x \\ f_y\ y + c_y \end{cases} \quad , \quad \text{with} \quad \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \boldsymbol{K}$$

Formally,

$$(\mathrm{X}, \mathrm{Y}, \mathrm{Z})^T \mapsto (f_x\ \mathrm{X/Z} + c_x, f_y\ \mathrm{Y/Z} + c_y)^T$$

where f_x, f_y and c_x, c_y values can be retrieved from the RAFFAELLO camera $\boldsymbol{K}$ matrix.

The sensor modeling used in this work, however, incorporates also radial and tangential distortions, which are typical for cameras.

It should be noted that most formulas found in literature use the inverse problem of determining the undistorted image, starting from a distorted acquisition. Our work is the exact opposite, requiring the emulation of camera distortions, in order to produce a realistic emulation.

The complete model of the map, incorporating the distortions, is [29]:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} = \begin{cases} f_x\ x_d + c_x \\ f_y\ y_d + c_y \end{cases}$$

Formally,

$$(\mathrm{X}, \mathrm{Y}, \mathrm{Z})^T \mapsto (f_x\ x_d + c_x, f_y\ y_d + c_y)^T$$

where x_d and y_d are the distorted quantities found as:

$$\begin{cases} x_d = x \cdot L(r) + \delta_x \\ y_d = y \cdot L(r) + \delta_y \end{cases}$$

The term $L(r)$ is the radial distortion contribution, while δ_x and δ_y are the tangential contributions.

Regarding radial distortion, it is based on the summation of harmonics, using a polynomial approximation using even powers of r (up to sixth order)[6]:

$$L(r) = 1 + k_1 r^2 + k_2 r^4 + k_3 r^6, \quad \text{with} \quad r = \sqrt{x^2 + y^2}$$

While the tangential distortion is modeled as:

$$\begin{cases} \delta_x = 2p_1 xy + p_2(r^2 + 2x^2) \\ \delta_y = p_1(r^2 + 2y^2) + 2p_2 xy \end{cases}$$

The coefficients k_1, k_2, k_3 and p_1, p_2 are provided as part of the RAFFAELLO camera intrinsic parameters, together with the matrix $\boldsymbol{K}$.

[6]OpenCV, Camera calibration With OpenCV. Available at `https://docs.opencv.org/4.x/d4/d94/tutorial_camera_calibration.html`

3 | Results

This last chapter presents the results of the optimization, to better understand how the solutions are displayed and reported in the Digital Twin version of the testbed. Results of the dataset acquisitions are also reported.

The given trajectory is a custom mission, designed in proximity of 25143 Itokawa (1998 SF36): the DART Lab already machined a mockup of the asteroid, with a length scale of $\delta_L = 1/669$. This mockup is completely covered by markers and a through hole is present, enabling both main faces to be mounted on the end effector of $GP12_2$: the remaining hole is sealed with a plug.

It is important noting how, this scale, is not well suited for the complete optimization of the given trajectory and a smaller scale shall be introduced to map bigger portions of it. The HIL solution is using the correct scale of the mockup, while, additional tests of the digital twin, have been carried out with a more suited scale of $\delta_L = 1/2000$. This scale is capable of mapping the completeness of the trajectory. The length scale discrepancies are the result of a renovation which is currently targeting the RAFFAELLO facility: indeed, the previous implementation of the testbed used a linear rail to move the base of the $GP12_1$, allowing to cover further distances and enabling to map bigger trajectories. This, however, is not a problem, since solutions have been found both for high-fidelity HIL tests and for simulation purposes.

Regarding the time scale, $\delta_T = 1/1000$ has been chosen to map single leg facility simulation of 30 s. Current HIL implementation takes more time, since each pose of the trajectory is requested as a single joint space target.

3.1. Digital Twin

The implementation of the optimization was carried out by creating a Digital Twin of the RAFFAELLO facility. Robots, payloads and environment have been modeled with high-fidelity to reproduce real mechanics.

This model is used to find a consistent solution to the given problem, while checking for

safety constraints. Tridimensional animations show the complete movements of robots and payloads, helping understanding how the real testbed would move in time.

The results of the MATLAB Digital Twin are shown below, where the main figures of the post processing of the optimization are reported and commented. The shown solution is $x_1 = 3.7028\,\text{rad}$, $x_2 = 0.0186\,\text{rad}\,\text{s}^{-1}$ and $x_3 = 0.5169$.

First of all, a snapshot of the facility is reported:

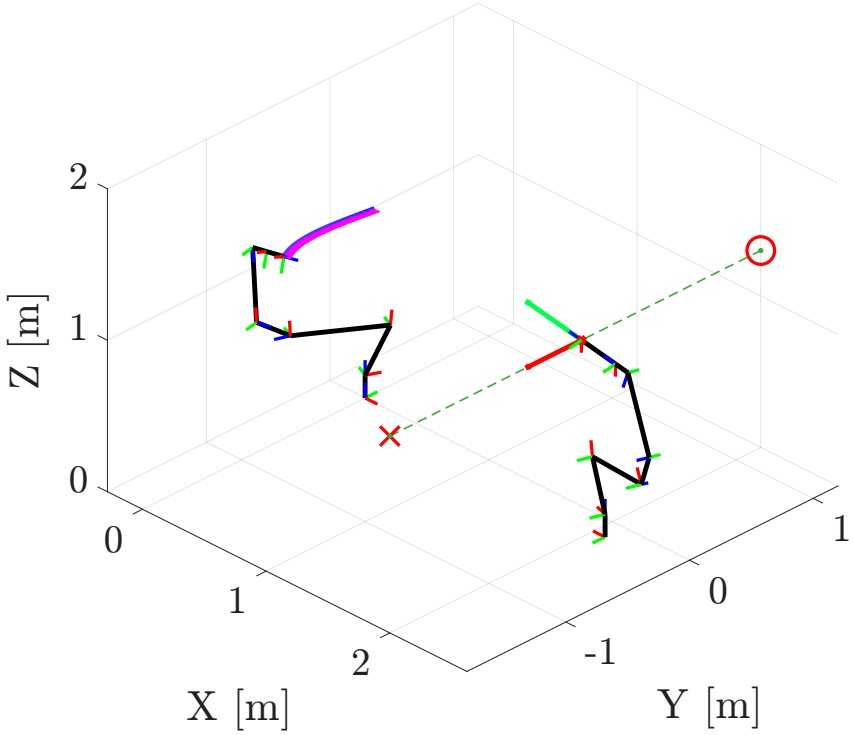


Figure 3.1: Facility solution configuration ($\{G\}$).

Figure 3.1 shows the initial configuration of the facility. Both robots are reported in their respective positions and orientations inside the safety cage, in the $\{G\}$ frame. The axis of the joints of the robot are reported with color coded arrows: red, green and blue for $\hat{\boldsymbol{e}}_1$, $\hat{\boldsymbol{e}}_2$ and $\hat{\boldsymbol{e}}_3$, respectively, while the lamp direction is denoted with the green dashed line (the red cross and circle represent the minimum and maximum lamp distance, respectively).

The red arrow is instead the Sun direction, seen by the target body: in the figure it is pointing the -y direction. While, the dig green arrow it the main axis of rotation of the target. Concerning the trajectory of the first robot, it is represented with a blue line, while the small pink arrows are the instantaneous directions of the $\hat{\boldsymbol{e}}_3$ axis of the camera payload.

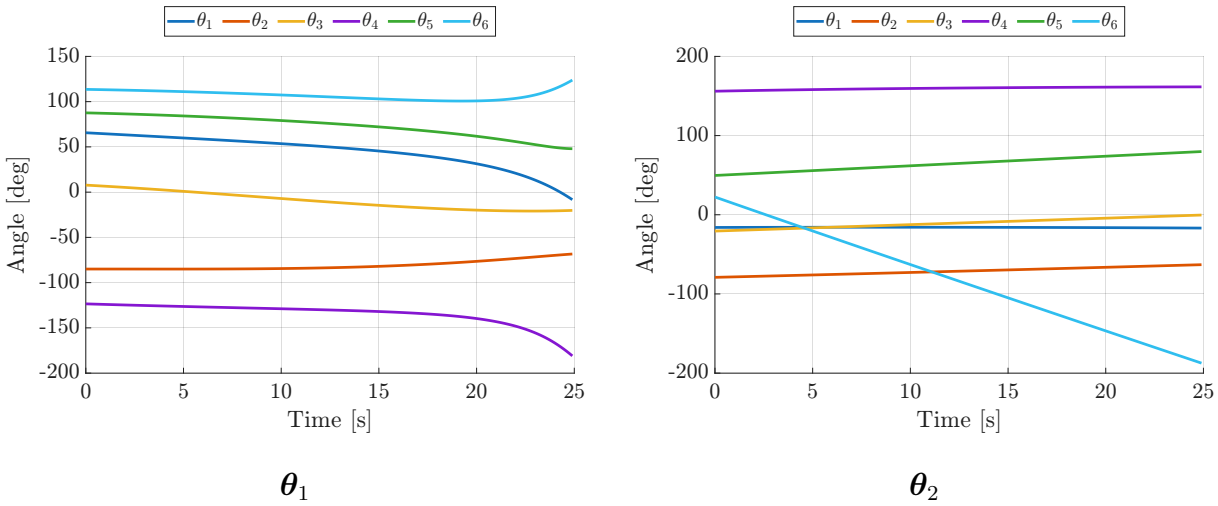


Figure 3.2: GP12$_1$ and GP12$_2$ $\boldsymbol{\theta}$ solutions.

The first $\boldsymbol{\theta}$ results can be seen in Figure 3.2, where all 6 joint solutions are displayed together, allowing for a quick look at the solution. This figure is useful to ensure no discontinuities or jumps are present in the solution.

A more in depth analysis of each joints is then presented in Figure 3.3 and Figure 3.4: these plots are used to verify the non violation of the $\boldsymbol{\theta}$ limits. These are important, since a small violation may happen, and a visual check is necessary. The theta limits are reported with the red, dashed lines.

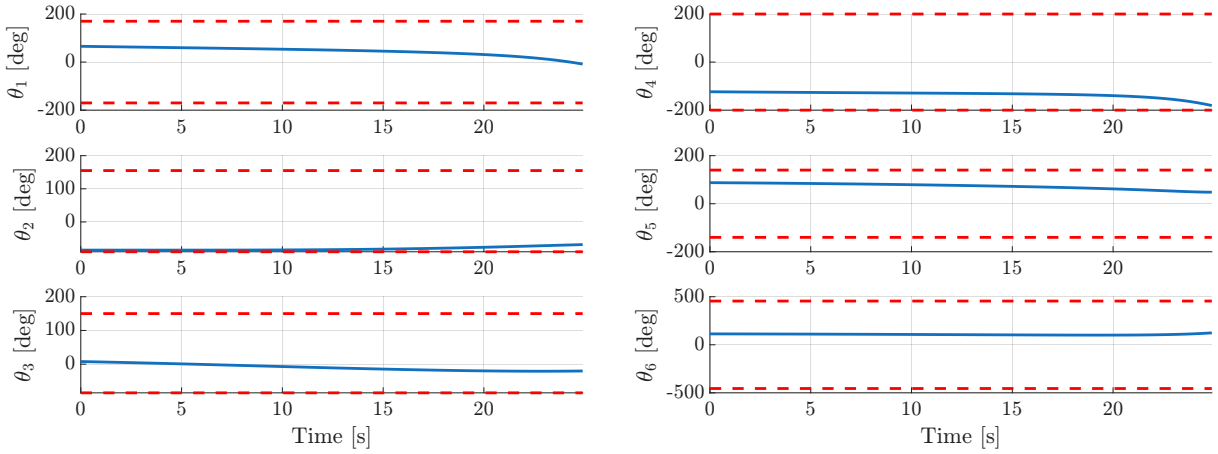


Figure 3.3: GP12$_1$ focus $\boldsymbol{\theta}_1$ solution.

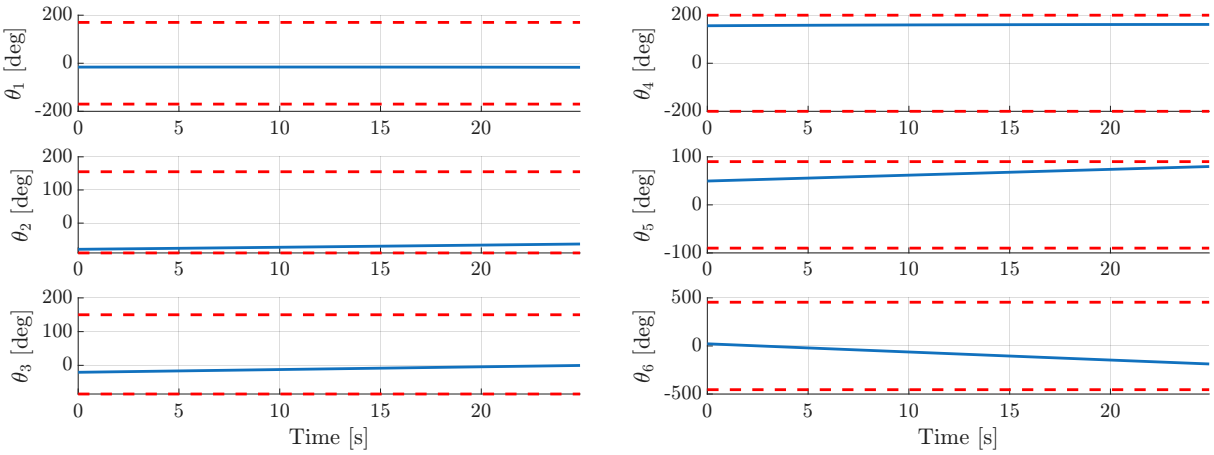


Figure 3.4: $GP12_2$ focus $\boldsymbol{\theta}_2$ solution.

One additional note, regarding the $\boldsymbol{\theta}$ limit plots: the red dashed lines refer to the GP12 specifications (Table 2.2 and Table 2.3), while the δ_θ values are not reported.

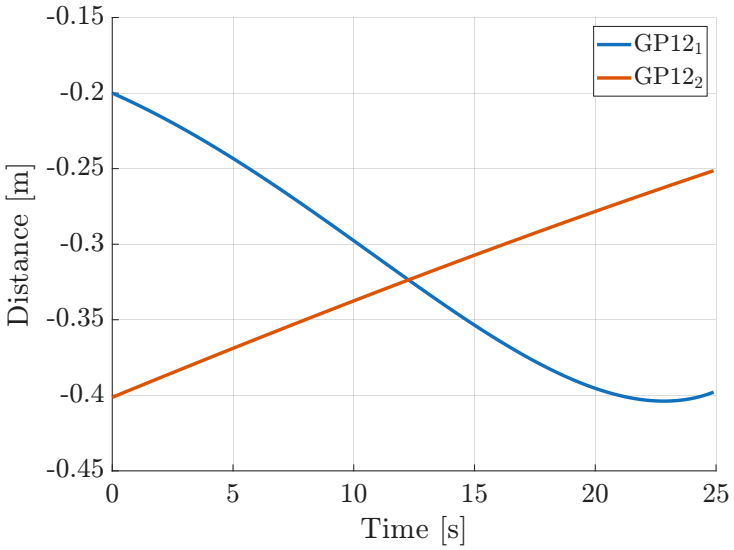


Figure 3.5: Reachability of $GP12_1$ and $GP12_2$.

Figure 3.5 shows the reachability margin, showing the distance from the maximum extension of the robots, as already discussed in section 2.1.3. It can be noted how the $GP12_1$ reaches the maximum feasible value of $-0.2\,\text{m}$ (imposed by δ_{distance}) at the initial time.

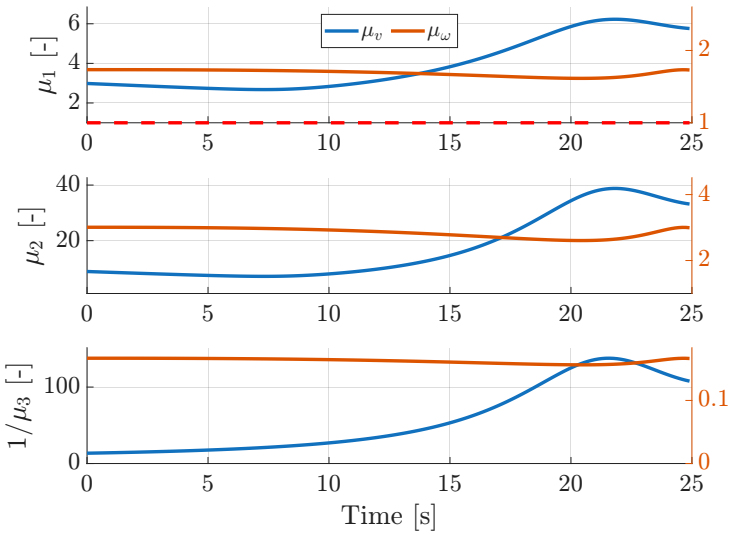


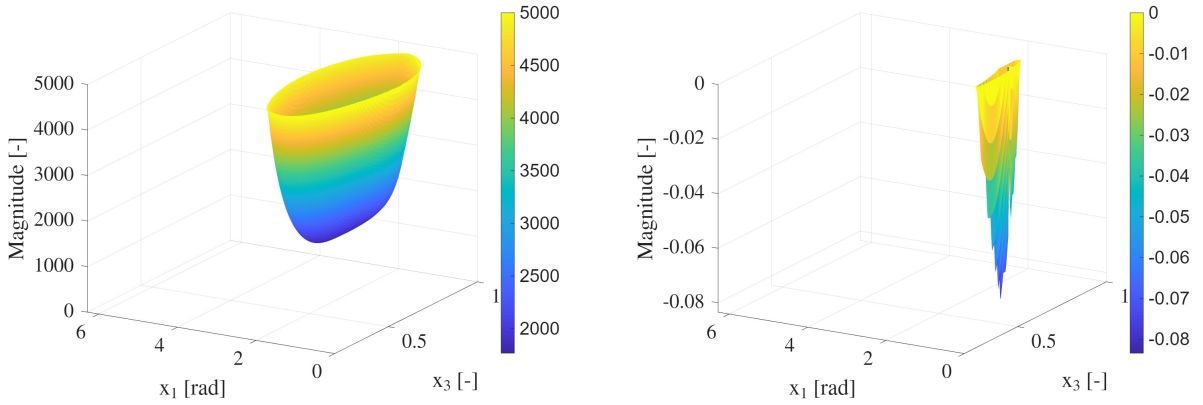
Figure 3.6: $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$ and $\boldsymbol{\mu}_3$ for GP12_1.

Regarding the last important plot, Figure 3.6 shows the evolution of $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$ and $\boldsymbol{\mu}_3$ for the first robot. These values are used in the cost function evaluation, and this plot is useful to determine the proximity to singularities.

The μ_1 and μ_2 values are expected to be high, since the convergence of the solution is affected by the nonlinear constraints small feasibility region. This phenomenon will be discussed in section 3.1.1.

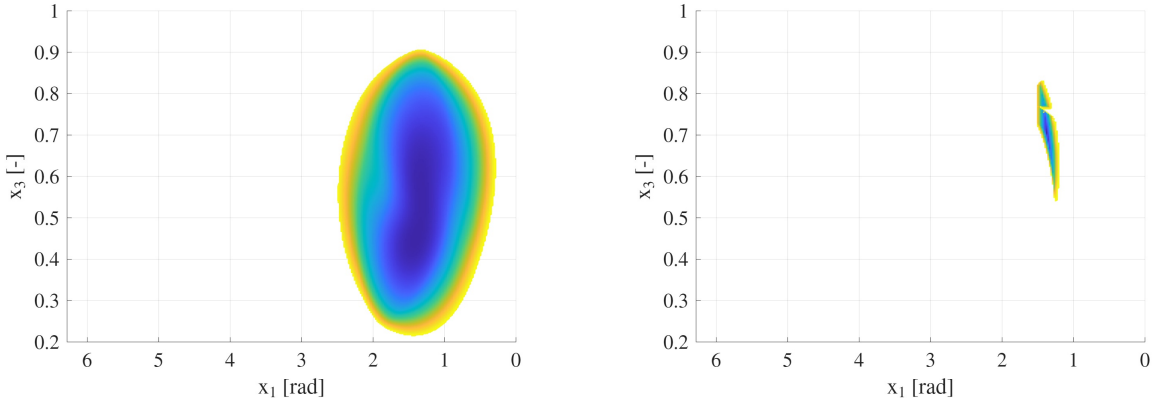
3.1.1. Cost Function and Constraints Analysis

An analysis of the shape of the cost function and nonlinear constraints domains is presented, to better understand the expected convergences.



Cost function Maximum of nonlinear constraints

Figure 3.7: $\delta_L = 1/2000$ ($x_2 = 0$): Cost function and nonlinear constraints contour plots.



Cost function Maximum of nonlinear constraints

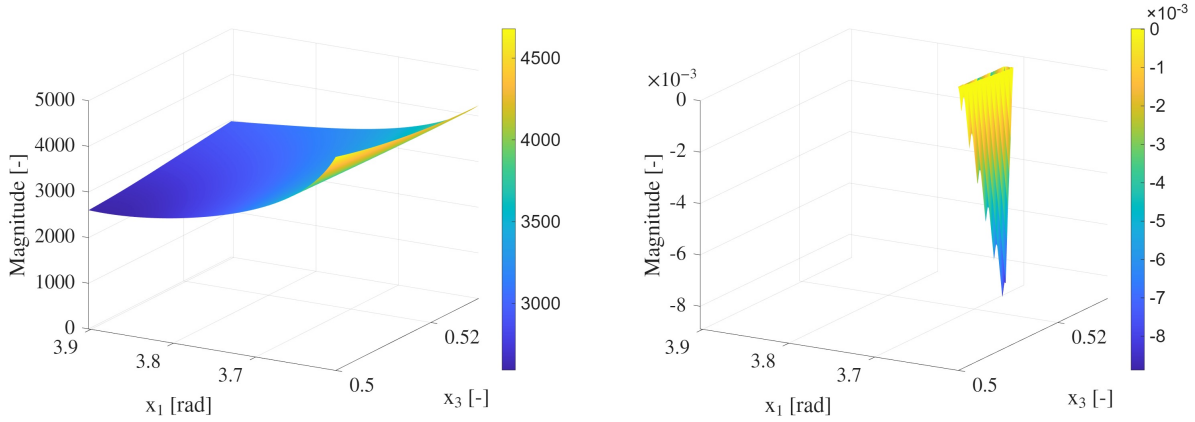
Figure 3.8: $\delta_L = 1/2000$ ($x_2 = 0$): Cost function and nonlinear constraints contour plots (top view).

Figure 3.7 and Figure 3.8 represent the cost function and the maximum value of the nonlinear constraints evaluation, given the solution reported on the axis. This solution is related to the $\delta_L = 1/2000$ scaling, which proved to be the most effective scale for the current RAFFAELLO configuration. Furthermore, x_2 is set to 0, since it is correlated to the biggest feasibility region.

The plots have been truncated above certain values to better highlight the behavior of the shawn quantities.

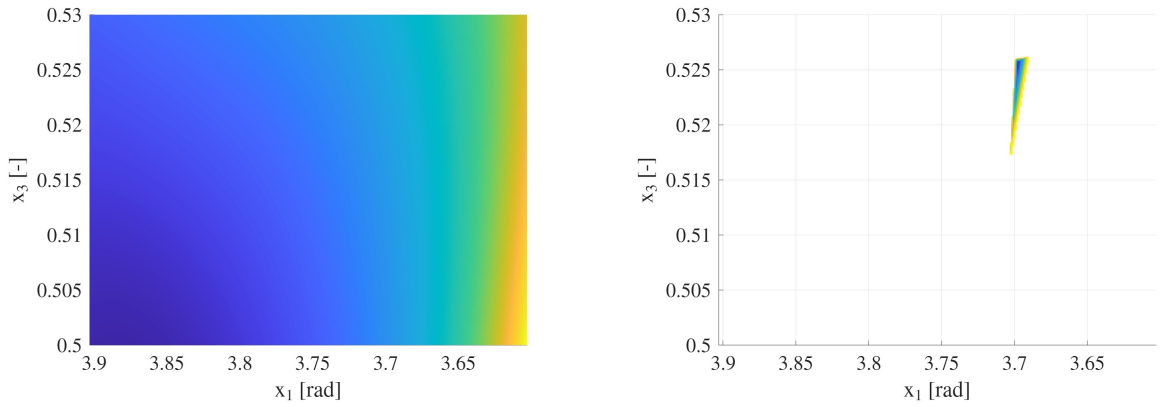
It can be noted how the cost function and the nonlinear constraints contour plots do intersect and the solution appears to be located close to the minimum of the cost function.

Analyzing now the $\delta_L = 1/669$ scaling problem, it can be noted how the feasible portions of the plots appear smaller. This result was expected, since the solutions are much harder to be found.



Cost function Maximum of nonlinear constraints

Figure 3.9: $\delta_L = 1/669$ ($x_2 = 0.0186$): Cost function and nonlinear constraints contour plots.



Cost function Maximum of nonlinear constraints

Figure 3.10: $\delta_L = 1/669$ ($x_2 = 0.0186$): Cost function and nonlinear constraints contour plots (top view).

As expected, the feasible portion of the domain is extremely small. However, the optimization is capable of finding the solution using the initial guess finder presented in section 2.3.3, demonstrating how its implementation can be considered to be robust.

3.1.2. Downsampling Results

As already mentioned in section 2.3.3, the optimization process was tested using the downsampling technique. This was done both to demonstrate faster convergence and to quantify the interpolation errors at points not included in the original optimization grid.

The results are generally consistent with those obtained for the nominal (full-resolution) problem, with the notable exception of higher errors at the newly interpolated points. This behavior is a direct consequence of the interpolation process itself, which introduces numerical inaccuracies. Several interpolation methods were tested, but the most accurate method proved to be the conversion of the attitude matrices to quaternions prior to pose interpolation. For the other quantities, spline interpolation proved to be the most precise. Unfortunately, even in this case the attitude errors reached orders of 1×10^{-6}, confirming that the main bottleneck remains the discretization of the problem.

Overcoming this issue, a simple strategy can be employed: the downsampled solution $\boldsymbol{x}^{\text{downsampled}}$ turns out to be extremely close to the full-resolution solution. This is expected, since the method is based on a geometrical approach and preserves the same boundary conditions. Thanks to this property, the $\boldsymbol{x}^{\text{downsampled}}$ solution can be directly employed for problems featuring a much larger number of points: the downsampled solution is evaluated on the complete (high-resolution) problem, and the nonlinear constraints are checked afterwards (with no violations expected in typical cases).

In this way, the same accuracy as solving directly the full-resolution problem is achieved, since no interpolation is being used: it is just an evaluation of the existing solution at additional discretization points. This extra evaluation step is computationally inexpensive.

Additional tests were performed using this downsampling technique with $\delta_L = 1/2000$, applied to a total of four trajectory legs. The main results are summarized in Table 3.1.

	Original		Downsampling			
Leg ID	**Time (s)**	**Points**	**Time (s)**	**Points**	**Time margin**	**Notes**
2,1	28.220	211	6.941	54	−75.40 %	
2,2	15.891	213	5.377	54	−66.16 %	same solution
5,2	6.385	106	3.313	54	−48.11 %	same solution
6,1	29.856	328	4.823	56	−83.84 %	same solution

Table 3.1: Downsampling results.

As shown in Table 3.1, the downsampling approach leads to substantial reductions in

computation time. When the downsampled solutions were subsequently validated by evaluating them on the full original (high-resolution) trajectory, complete feasibility was confirmed in all cases. The first leg *2,1* exhibits only minor deviations from the original solution:

Leg 2,1	Original solution	Downsampling solution
x_1	1.243 38 rad	1.242 15 rad
x_2	0.009 48 rad s^{-1}	0.009 52 rad s^{-1}
x_3	0.645 59	0.644 63

Table 3.2: Downsampling results (2,1 leg solution).

3.2. Validation Results

Focusing now on the validation results, the first analysis regards the comparison between the result of the IVK and the IK. Starting from the $\dot{\boldsymbol{\theta}}$ solution, it is integrated to obtain $\boldsymbol{\theta}$: this value is then compared to the result of the IK, which corresponds to the reference.

Figure 3.11, Figure 3.12 and Figure 3.13 show the comparison between the optimization and the numerical solutions.

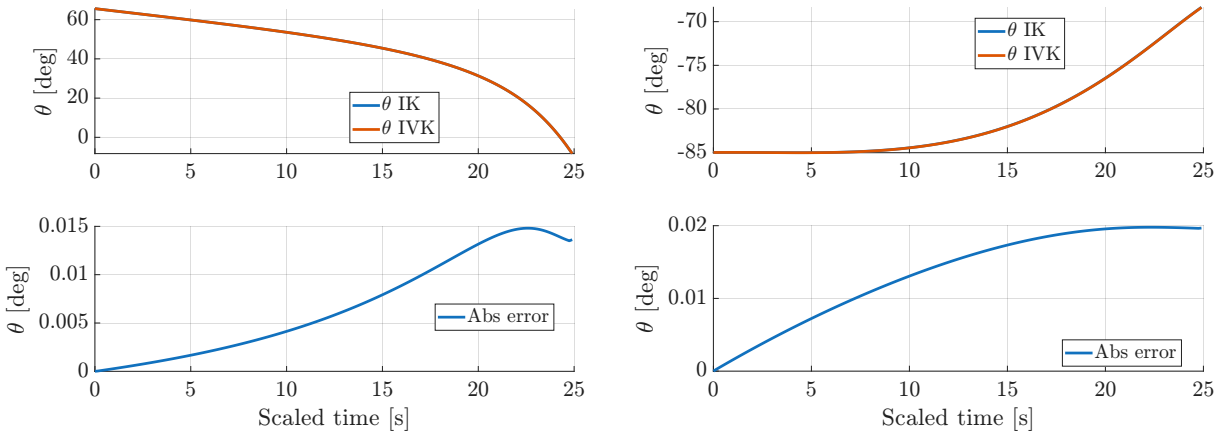


Figure 3.11: GP12$_1$: θ_1 and θ_2 solutions and errors.

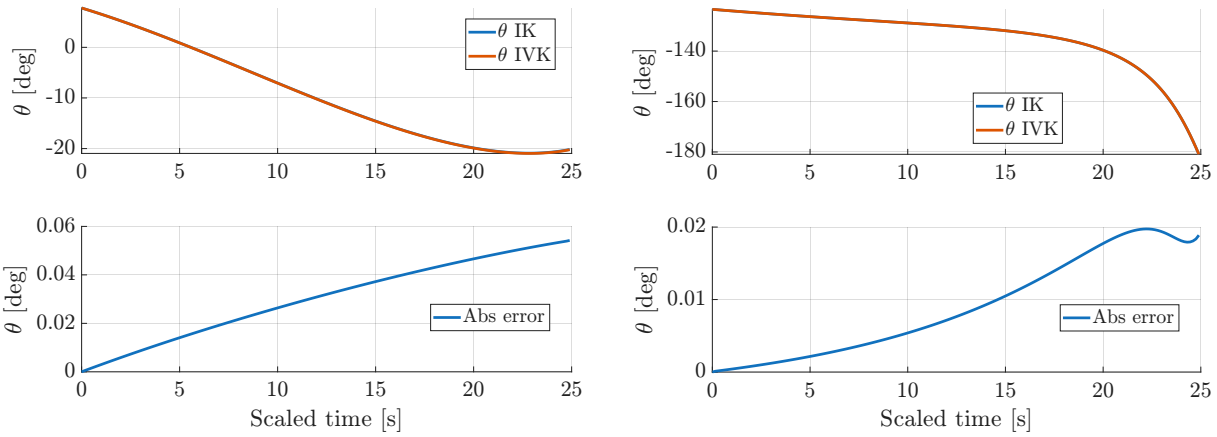


Figure 3.12: GP12$_1$: θ_3 and θ_4 solutions and errors.

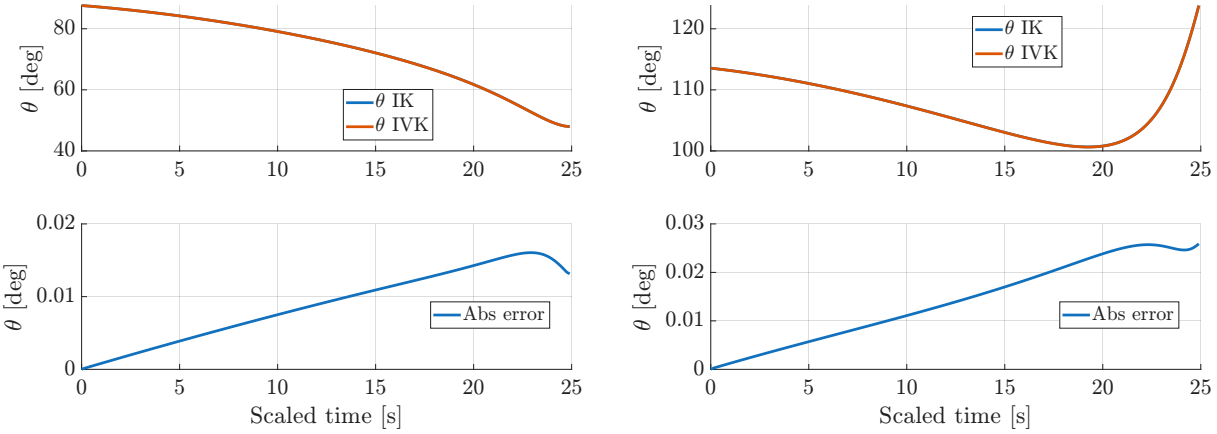


Figure 3.13: GP12$_1$: θ_5 and θ_6 solutions and errors.

It can be noted how the errors are small if compared to the magnitude of the solution and the overall behavior is correctly represented. This proves a correct implementation of the Jacobian. Furthermore, no discontinuities are visible in the analyzed data.

Comparing now the results of the second GP12, the following plot shows the same behavior:

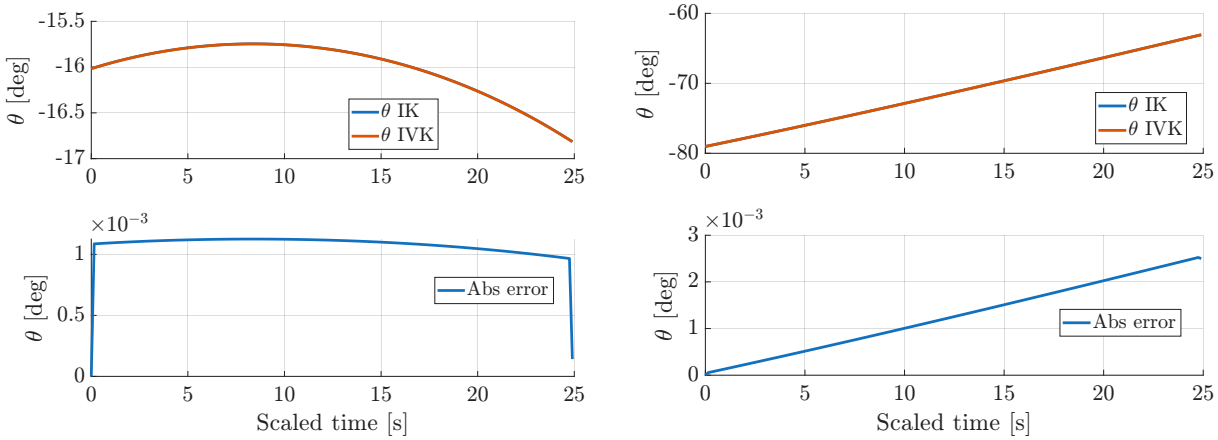


Figure 3.14: GP12$_2$: θ_1 and θ_2 solutions and errors.

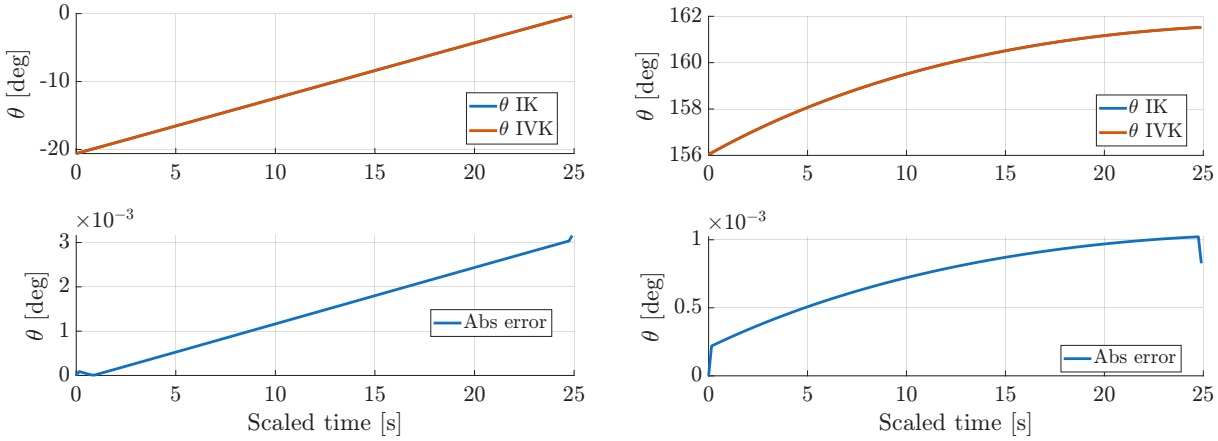


Figure 3.15: GP12$_2$: θ_3 and θ_4 solutions and errors.

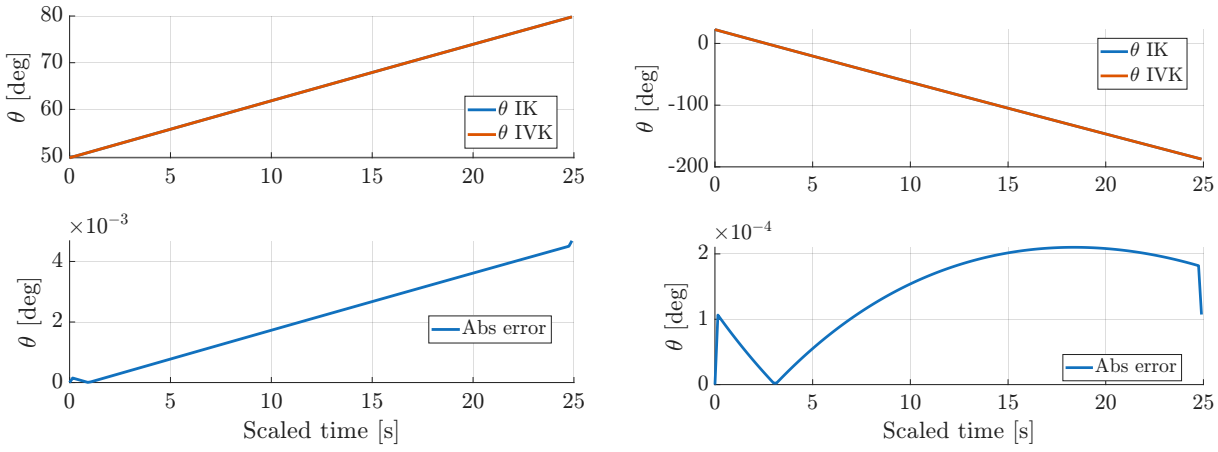


Figure 3.16: GP12$_2$: θ_5 and θ_6 solutions and errors.

The errors of the GP12$_2$ analysis are similar to the GP12$_1$, thus validating the correctness

of the Jacobian.

Concerning the validation of the trajectories, few plots are presented below, where the errors are expected to be extremely small. Starting from ${}^{T}\boldsymbol{r}_C$, Figure 3.17 shows both the optimization solution and the input trajectory. These values are then compared and the error is presented in the following figure.

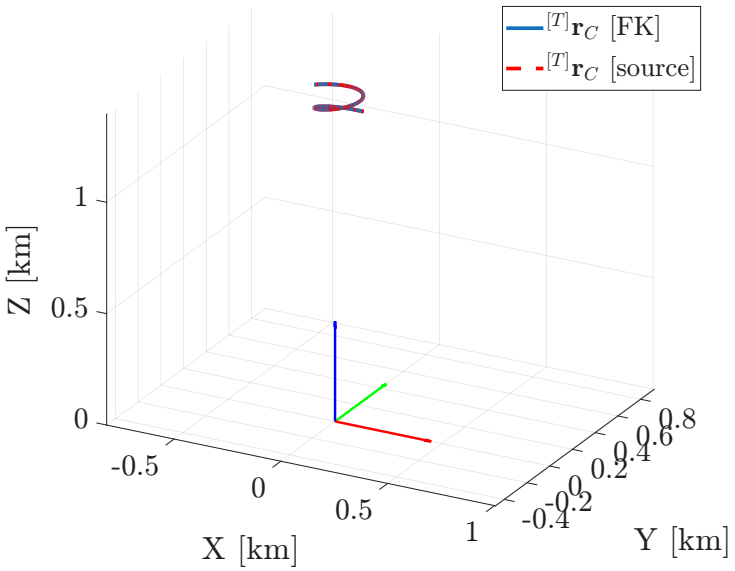


Figure 3.17: ${}^{T}\boldsymbol{r}_C$.

Figure 3.17 shows the $\{T\}$ frame, where its axis are drawn with red, green and blue (RGB) arrows, respectively symbolizing ${}^{T}\hat{\boldsymbol{e}}_1$, ${}^{T}\hat{\boldsymbol{e}}_2$ and ${}^{T}\hat{\boldsymbol{e}}_3$.

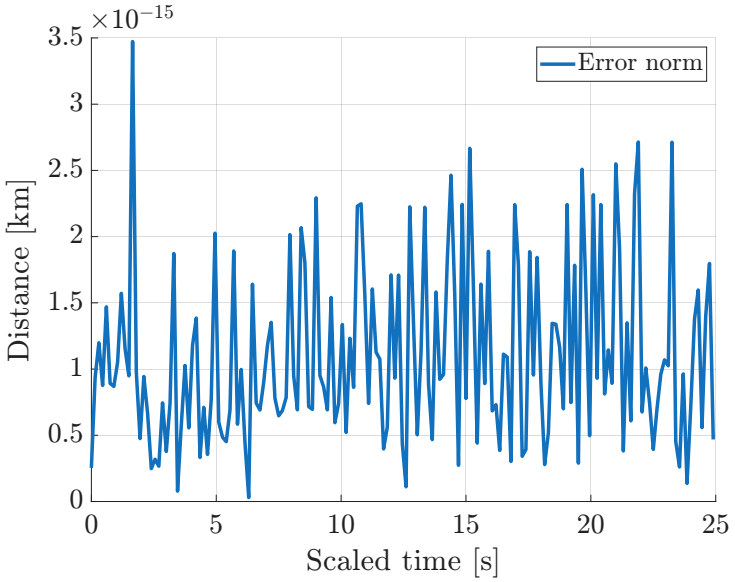


Figure 3.18: ${}^{T}\boldsymbol{r}_C$ error.

The difference between the recreated trajectory and the reference can be checked in Figure 3.18, where the magnitude of the errors is extremely low, corresponding to a machine precision error. This proves the optimization to be capable of representing the trajectory in the $\{T\}$ without introducing errors.

Following with the validation of the velocity ${}^{T}\boldsymbol{v}_C$, Figure 3.19 shows the comparison of the individual components with the reference values.

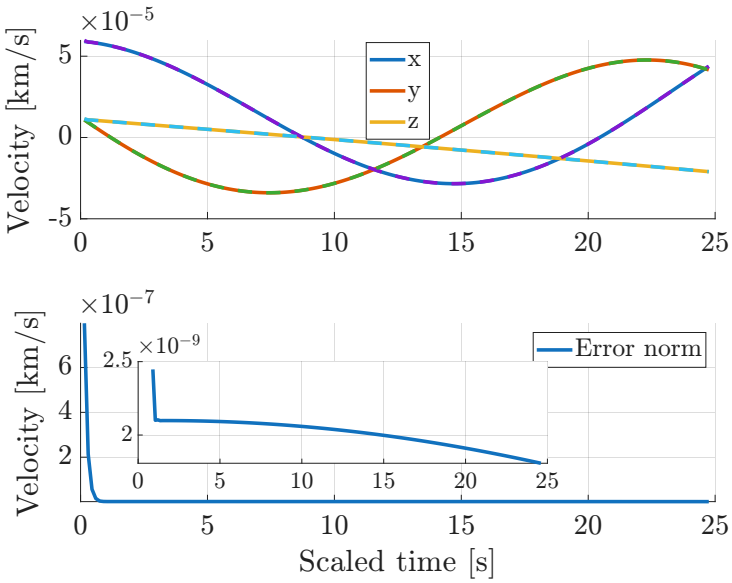


Figure 3.19: ${}^{T}\boldsymbol{v}_C$ components comparison and error.

It can be seen how the errors shown in Figure 3.19 are not similar to the previous analysis: this is given by the approximation introduced in the transportation of the velocities, across the different rotating reference frames, which is performed thanks to the discrete derivatives of the rotation matrices. This result was expected, but its errors are within reasonable values. Furthermore, an initial spike is visible, reflecting the higher errors typical of the less accurate forward/backward finite difference scheme applied at the boundaries.

It is important reminding how these values are only being used for the V&V, since no task space velocities will ever be used to simulate the map in the RAFFAELLO facility. The important quantities for this robotic implementation are the $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$ solutions. The first is derived with extremely small errors (compared to the mathematical model), while the second can be retrieved numerically from the first one: this ensures a smaller error if compared to the result of the IVK. However, for computational costs and for Jacobian validation, the IVK is still used inside the *core-mapper* function, since its results are perfectly valid for the nonlinear constraints evaluation.

Analyzing now the Sun versor ${}^{T}\hat{\boldsymbol{r}}_{sun}$, its value is compared in the $\{T\}$ frame.

Figure 3.20 shows how the direction of the Sun is coherent with the reference values, by calculating the norm error between the two values, while Figure 3.21 proves the phase angle of the spacecraft (camera) to be unaltered.

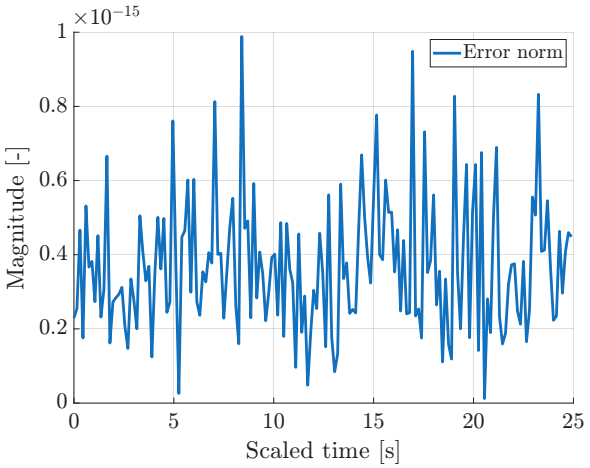


Figure 3.20: ${}^{T}\hat{\boldsymbol{r}}_{sun}$ error.

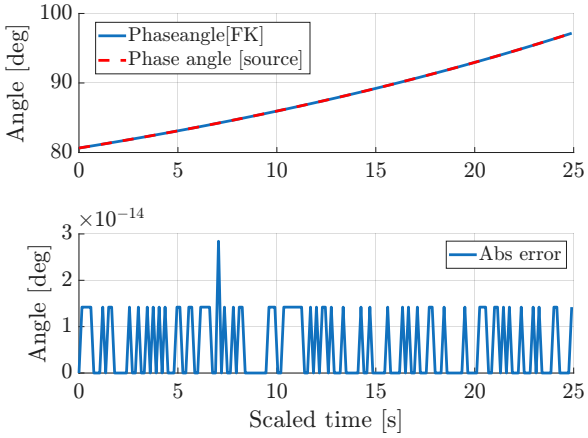
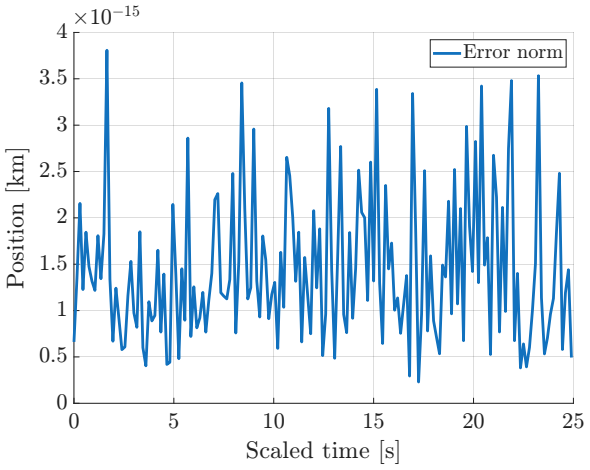
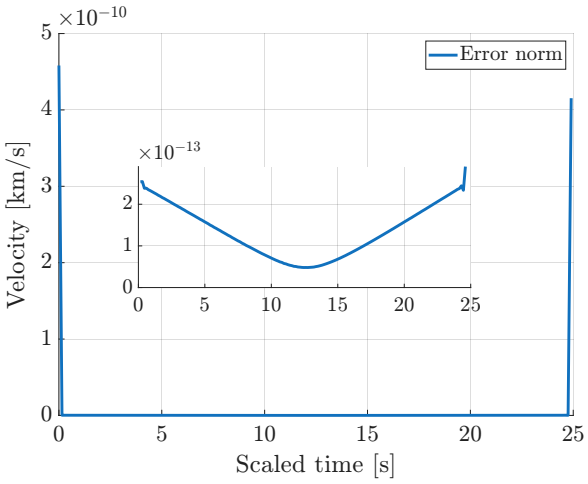


Figure 3.21: $\{T\}$ Sun phase angle.

The second and last analysis of the trajectory validation consists in calculating the same quantities reported above, this time in the $\{N\}$ frame. Starting from the position and velocity vectors:



${}^{N}\boldsymbol{r}_{C}$ error



${}^{N}\boldsymbol{v}_{C}$ error (with focus)

Figure 3.22: ${}^{N}\boldsymbol{r}_{C}$ and ${}^{N}\boldsymbol{v}_{C}$ errors.

The same behavior presented for the $\{T\}$ can be noted in this new analysis: the errors of ${}^{N}\boldsymbol{r}_{C}$ are extremely small, whereas the velocities present two orders of magnitude of differ-

ence. This is caused by the same problem discussed above: transposition across rotating reference frame, using discrete approximations. Higher errors at the boundaries result from the use of forward/backward finite difference schemes, which are less accurate than the centered differences applied at the interior points. This behavior is consistent with that observed in Figure 3.19. The overall errors, however, are extremely low, indicating a correct implementation of Equation 2.21 and the effects of the discretization are smaller if compared to Figure 3.19.

The results have been compared with a MATLAB built in function which can be used to retrieve the angular velocities of the attitudes expressed in quaternions, but, the results, showed little to no improvements.

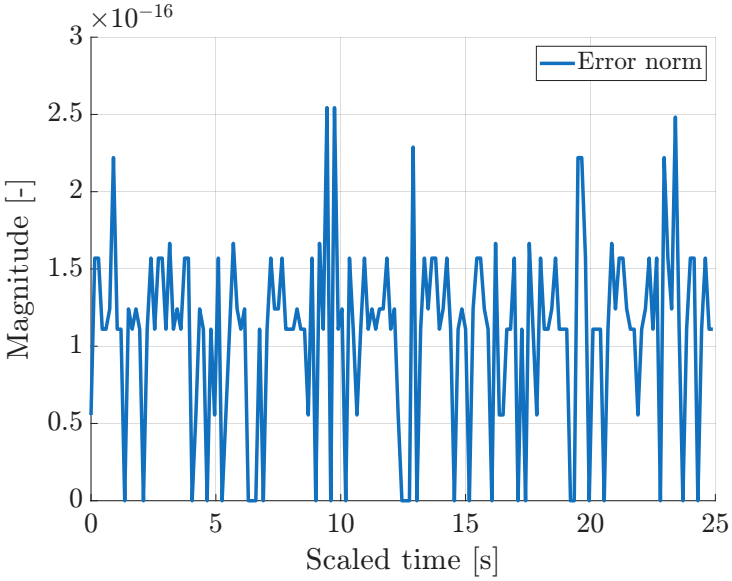


Figure 3.23: ${}^{N}\hat{\boldsymbol{r}}_{sun}$ error.

Figure 3.23 is self-explanatory, and the errors are as expected.

In addition, a comparison of the attitude matrices is presented here. The errors have been computed as the norm of the difference between the reconstructed matrices and the reference ones. The corresponding results are shown in Figure 3.24 and Figure 3.25, where, as anticipated, the errors remain on the order of machine precision.

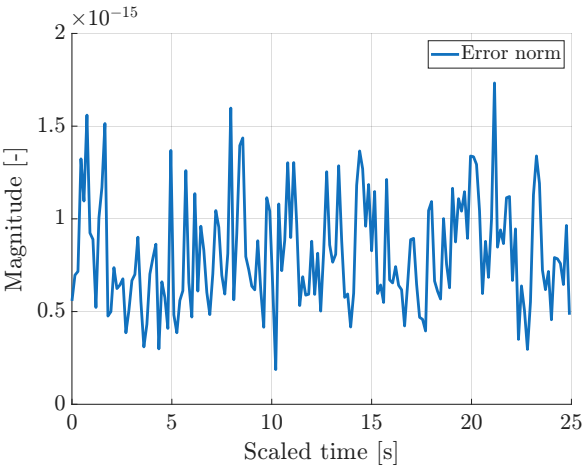


Figure 3.24: $\boldsymbol{R}_{N/C}$ error.

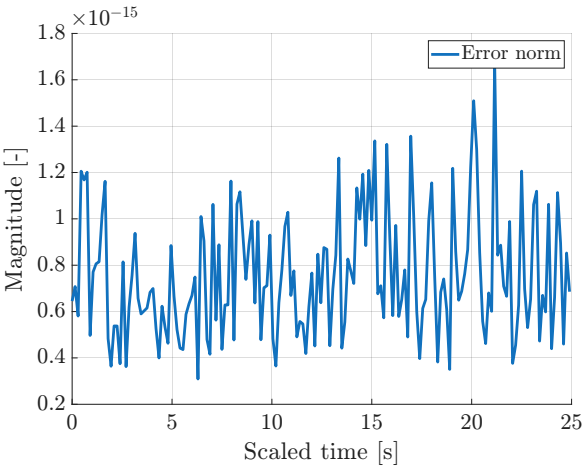


Figure 3.25: $\boldsymbol{R}_{N/T}$ error.

Last validation results regard the camera emulation: first of all, a set of fixed points have to be defined in the correct $\{T\}$ frame, since they will act as reference points for the comparison. Figure 3.26 show the selected elements.

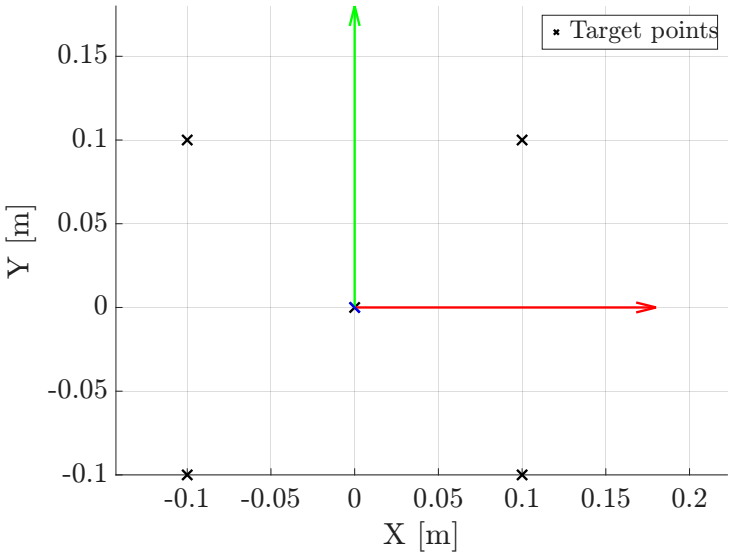


Figure 3.26: $\{T\}$ target points.

The trajectory is projected in the in the $\{T\}$ frame, where pointing and points position can be evaluated.

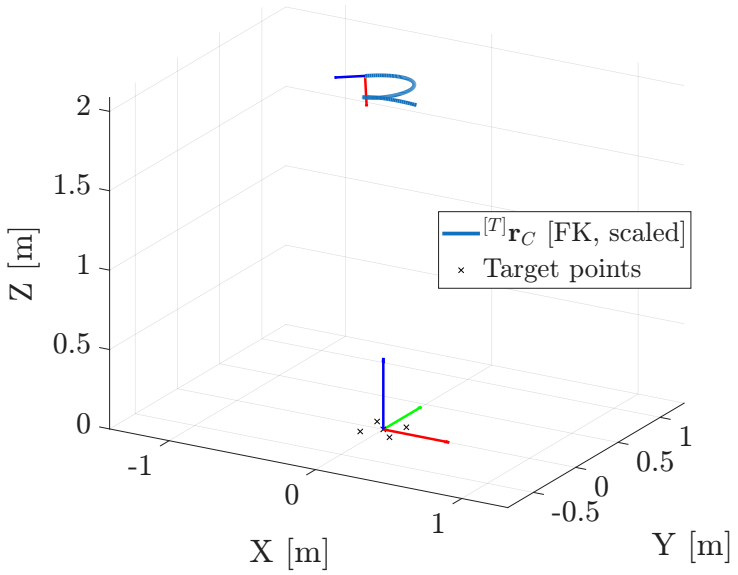


Figure 3.27: $\{T\}$ trajectory.

Otherwise, by representing the points in the $\{C\}$ frame, the quantities needed for the sensor emulation can be retrieved. In Figure 3.28, the point paths are plotted and the errors with the original reference frames transformations are compared.

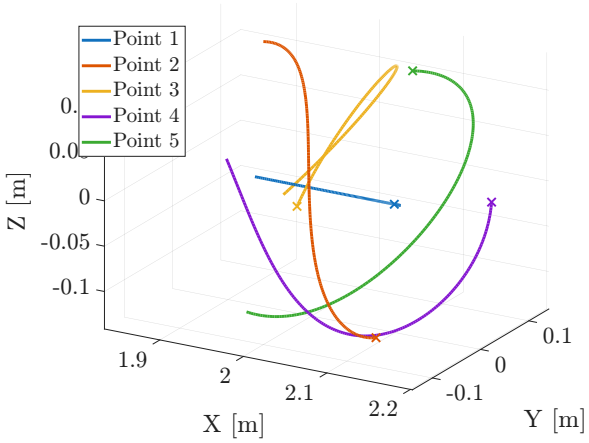


Figure 3.28: $\{C\}$ target points.

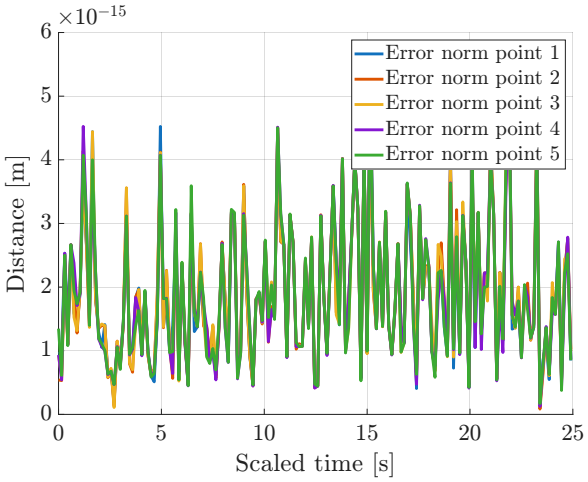


Figure 3.29: $\{C\}$ target points error.

The errors are extremely small, proving its values to be correctly represented by the optimization. The same analysis is presented for the modeled camera sensor, where the points and the errors are reported in pixel.

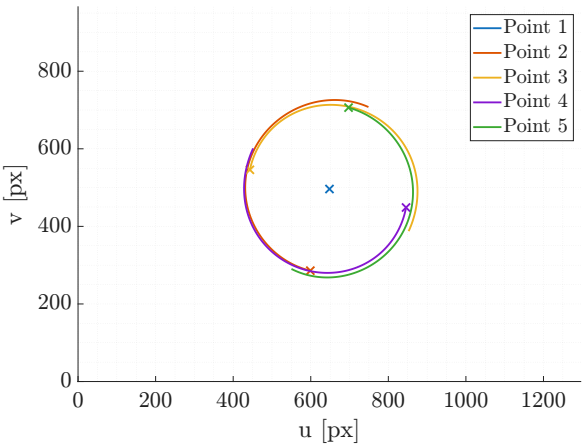


Figure 3.30: Modeled camera sensor.

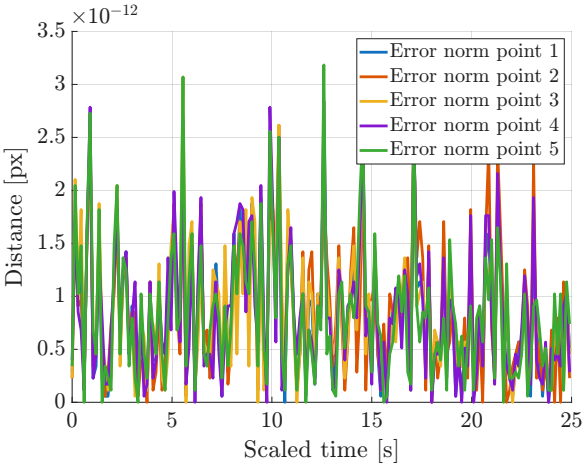


Figure 3.31: Modeled camera sensor, target points error.

Figure 3.30 shows the path of the camera sensor projected points. Each point is color coded, and the initial position is marked with a cross symbol of the same color. The pattern of the points has been created with a central first point: it can be seen how its position never changes, proving a correct alignment of the pointing. This can be appreciated both in Figure 3.30 and in Figure 3.28.

Figure 3.31 instead shows the normalized errors of the projected points. It is clear how the result would not be appreciated in a real life camera sensor, given the extremely small differences.

3.3. Hardware In the Loop tests

Regarding HIL tests, the RAFFAELLO facility can be used to generate datasets of a given target mockup. The following data have been acquired with 25143 Itokawa (1998 SF36), as shown in Figure 3.32 and in Figure 3.33.

Figure 3.32: $GP12_1$ and $GP12_2$ inside RAFFAELLO facility.

Figure 3.33: RAFFAELLO facility.

The current HIL implementation allow to define a finite set of poses, which are fed to the Yaskawa controller. The joint solutions are then sent to the robots, that reach the desired configuration, where, the camera, captures the specific pose image.

This implementation allows to correctly reproduce a finite set of poses, but lacks of dynamics emulation.

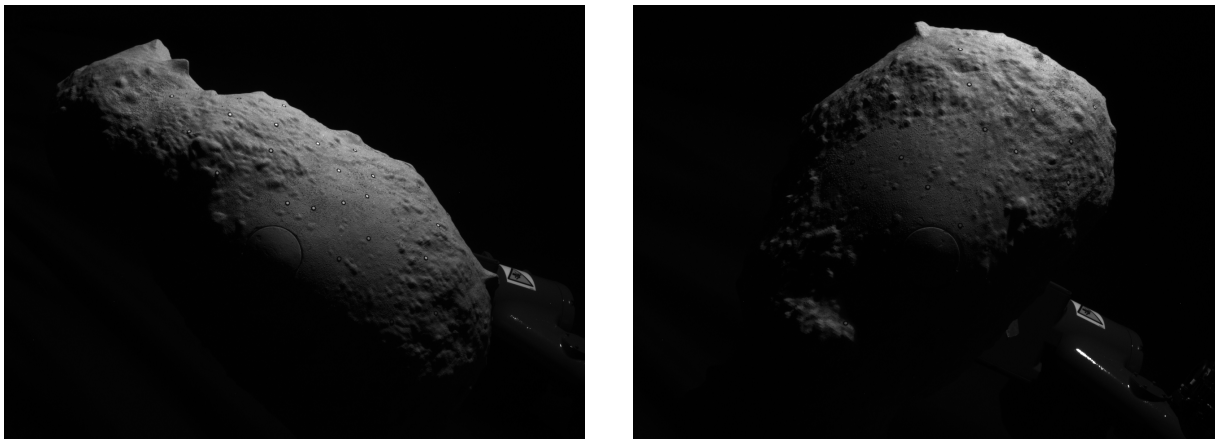


Figure 3.34: Frames 0075/0167 and 0155/0167 of dataset acquisition.

Concluding the results analysis, Figure 3.34 shows a dataset acquisition test using the mapped trajectory. The frames are unprocessed: the robot arm and plug are clearly visible. In this case, post-processing is usually required before using images for navigation algorithms testing.

4 | Conclusions and Future Work

Given all the results presented in chapter 3, it is clear how the mapping technique is capable of representing the real dynamic behavior of a real mission. The scaled environment allow to reduce the problem to a confined, testbed, space, where the evolution of the problem is reconstructed using high torque actuators.

Errors related to position vectors and attitudes are all in the order of numerical precision errors: these values were expected, since the validation steps present only matrix multiplications for such elements. The transformation between the task space solution and the joint space solution is similar: a robotically feasible solution introduces similar errors, for the very same reason.

Velocities introduce a particular challenge in the analysis, as transitions between rotating reference frames introduce discretization errors from finite-difference velocity transposition. These errors, now reduced to the order of $1 \times 10^{-13}\,\mathrm{km\,s^{-1}}$, remain extremely small in absolute terms and negligible compared to the magnitude of the nominal relative velocities. Although they are comparatively larger than the position reconstruction errors, as expected from the discretized nature of the input trajectories and the finite sampling of angular velocity vectors, they do not compromise the overall accuracy of the mapping methodology.

Since the higher errors of the velocities come from a discretization problem, and not from an implementation error or incorrect assumptions, the mapping technique can still be validated, remembering that the discrepancies with the reference values are given by post processing inaccuracies.

Last important remark, these errors are typical for ideal trajectories, where the digital twin employed do not model additional errors or misalignment problems, typical for HIL testbeds, where the mathematical solution has to face the reality of physical hardware. It is worth emphasizing that the presented results are intended to validate the developed mapping technique under ideal conditions. No additional perturbations have been added to the simulation, since those analysis are not related to the scope of the thesis.

Regarding the optimization, no discontinuities are found in the $\boldsymbol{\theta}$ solution, proving how the implementation of the IK is consistent with the singular solution assumptions. Furthermore, these values proved to exactly replicate the real map between task and joint space, since IK and FK can be combined to reproduce the same original result: this is achieved by solving the problem with a trigonometric approach, since no iterative procedures are followed.

Concluding, various changes of frames and values relating misalignment have been used during the developed of the thesis, proving how the Digital Twin model can adapt to those changes, aligning with DART Lab models, which have been used to compare the solutions.

Among the future steps, one priority is to enable the reproduction of time-parameterized trajectories in the RAFFAELLO testbed, since the current configuration only allows to represent discretized poses. Several implementations suggest how the YRC1000 controller can be used to control the robots in joint space velocities, allowing to reproduce dynamic solutions. This would allow to completely reproduce the mapped trajectories and take advantage of the dynamically consistent map.

Bibliography

[1] M. Abrahamson, A. Ardito, D. Han, R. Haw, B. Kennedy, N. Mastrodemos, S. Nandi, R. Park, B. Rush, and A. Vaughan. Dawn orbit determination team: Trajectory modeling and reconstruction processes at vesta. volume 148, 02 2013.

[2] P. Antreasian, S. Chesley, C. Helfrich, T. Wang, W. Owen Jr, J. Miller, J. Bordi, and B. Williams. Near shoemaker's low altitude operations at eros. In *International Symposium on Space Flight Dynamics, Pasadena, CA*, volume 3, 2001.

[3] P. G. Antreasian, C. D. Adam, K. Berry, J. Geeraert, K. M. Getzandanner, D. Highsmith, J. M. Leonard, E. J. Lessac-Chenen, A. H. Levine, J. V. McAdams, et al. Osiris-rex proximity operations and navigation performance at bennu. In *AIAA SCITECH 2022 Forum*, page 2470, 2022.

[4] E. Bates, Z. Ahmed, A. G. Peretz, P. F. Huc, A. Rizza, S. Y. W. Low, T. Bell, G. Zin, and S. D'Amico. Digital and robotic twinning for validation of proximity operations and formation flying. In *Proceedings of the 48th Annual AAS Guidance, Navigation & Control Conference*, AAS 26-148, Breckenridge, Colorado, January 30–February 4 2026. American Astronautical Society.

[5] H. Benninghoff, F. Rems, E.-A. Risse, and C. Mietner. European proximity operations simulator 2.0 (epos)-a robotic-based rendezvous and docking simulator. *Journal of large-scale research facilities JLSRF*, 2017.

[6] J. S. Dai. Euler–rodrigues formula variations, quaternion conjugation and intrinsic connections. *Mechanism and Machine Theory*, 92:144–152, 2015.

[7] J. Denavit and R. Hartenberg. Notation for lower-pair mechanisms based on matrices. *A. Kinematic, ASME Journal of Applied Mechanics*, 22:215–221, 01 1995. doi: 10.1115/1.4011045.

[8] D. W. Dunham, J. V. McAdams, and R. W. Farquhar. Near mission design. *Johns Hopkins APL technical digest*, 23(1):18–33, 2002.

[9] L. Euler. Nova methodus motum corporum rigidorum degerminandi. *Novi commentarii academiae scientiarum Petropolitanae*, pages 208–238, 1776.

[10] C. Giordano, F. Topputo, S. Campagnola, M. Casasco, et al. Enhancing autonomy for close-proximity operations: the msca-funded project castor. In *75th International Astronautical Congress (IAC 2024)*, pages 1–5, 2024.

[11] B. Godard, F. Budnik, P. Munoz, T. Morley, and V. Janarthanan. Orbit determination of rosetta around comet 67p/churyumov-gerasimenko. In *Proceedings 25th International Symposium on Space Flight Dynamics–25th ISSFD, Munich, Germany*, 2015.

[12] D. Han. Orbit transfers for dawn's vesta operations: navigation and mission design experience. 2012.

[13] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[14] S. E. Hawkins III, E. H. Darlington, S. L. Murchie, K. Peacock, T. J. Harris, C. B. Hersman, M. J. Elko, D. T. Prendergast, B. W. Ballard, R. E. Gold, et al. Multispectral imager on the near earth asteroid rendezvous mission. *Space Science Reviews*, 82(1):31–100, 1997.

[15] H. Krüger, S. Theil, M. Sagliano, and S. Hartkopf. On-ground testing optical navigation systems for exploration missions. In *9th ESA Conference on Guidance, Navigation & Control Systems*, 2014.

[16] D. Lauretta, S. Balram-Knutson, E. Beshore, W. Boynton, C. Drouet d'Aubigny, D. DellaGiustina, H. Enos, D. Golish, C. Hergenrother, E. Howell, et al. Osiris-rex: sample return from asteroid (101955) bennu. *Space Science Reviews*, 212(1):925–984, 2017.

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

[18] N. Mastrodemos, B. Rush, A. Vaughan, and W. Owen. Optical navigation for the dawn mission at vesta. In *23rd International Symposium on Space Flight Dynamics, Pasadena, CA*, volume 29, 2012.

[19] R. M. Murray, Z. Li, and S. S. Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[20] NASA JPL. Small-body database lookup. `https://ssd.jpl.nasa.gov/tools/sbdb_lookup.html#/?des=a4`, checked in 2025.

[21] M. Olivares-Mendez, M. R. Makhdoomi, B. C. Yalçın, Z. Bokal, V. Muralidharan, M. O. Del Castillo, V. Gaudilliere, L. Pauly, O. Borgue, M. Alandihallaj, et al. Zero-g lab: A multi-purpose facility for emulating space operations. *Journal of Space Safety Engineering*, 10(4):509–521, 2023.

[22] T. H. Park, J. Bosse, and S. D'Amico. Robotic testbed for rendezvous and optical navigation: Multi-source calibration and machine learning use cases. *arXiv preprint arXiv:2108.05529*, 2021.

[23] J. Paul, F. Kirchner, I. Ahrns, and J. Sommer. Robotics rendezvous and capture test facility inveritas. In *Proceedings of 12th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2014),(Montreal, Canada)*, 2014.

[24] J. Paul, A. Dettmann, B. Girault, J. Hilljegerdes, F. Kirchner, I. Ahrns, and J. Sommer. Inveritas: a facility for hardware-in-the-loop long distance movement simulation for rendezvous and capture of satellites and other autonomous objects. *Acta Astronautica*, 116:1–24, 2015.

[25] D. L. Pieper. *The kinematics of manipulators under computer control*. Stanford University, 1969.

[26] M. D. Rayman. Dawn at ceres: The first exploration of the first dwarf planet discovered. *Acta Astronautica*, 194:334–352, 2022.

[27] C. Raymond, R. Jaumann, A. Nathues, H. Sierks, T. Roatsch, F. Preusker, F. Scholten, R. Gaskell, L. Jorda, H.-U. Keller, et al. The dawn topography investigation. *Space science reviews*, 163(1):487–510, 2011.

[28] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: modelling, planning and control*. Springer, 2009.

[29] Z. Tang, R. G. Von Gioi, P. Monasse, and J.-M. Morel. A precision analysis of camera distortion models. *IEEE Transactions on Image Processing*, 26(6):2694–2704, 2017.

[30] J. Veverka, J. Bell III, P. Thomas, A. Harch, S. Murchie, S. Hawkins III, J. Warren, H. Darlington, K. Peacock, C. Chapman, et al. An overview of the near multispectral imager-near-infrared spectrometer investigation. *Journal of Geophysical Research: Planets*, 102(E10):23709–23727, 1997.

[31] M. Wilde, C. Clark, and M. Romano. Historical survey of kinematic and dynamic spacecraft simulators for laboratory experimentation of on-orbit proximity maneuvers. *Progress in Aerospace Sciences*, 110:100552, 2019.

[32] B. G. Williams. Technical challenges and results for navigation of near shoemaker. *Johns Hopkins APL technical digest*, 23(1):34–45, 2002.

[33] D. A. Williams, D. L. Buczkowski, S. C. Mest, J. E. Scully, T. Platz, and T. Kneissl. Introduction: The geologic mapping of ceres. *Icarus*, 316:1–13, 2018.

[34] S. Woicke and H. Krüger. Hardware-in-the loop testing of crater navigation system for lunar landing. In *AIAA SciTech 2024 Forum*, page 0312, 2024.

[35] *MOTOMAN-GP12/AR1440 INSTRUCTIONS*. YASKAWA, 7 2016. HW1484060.

List of Figures

List of Tables

List of Symbols

Variable	Description	SI unit
$\boldsymbol{\delta}_{\theta}$	θ constraint margin	$^{\circ}$
$\boldsymbol{\delta}_{\dot{\theta}}$	$\dot{\theta}$ constraint margin	rad s
$\boldsymbol{\delta}_{\text{distance}}$	Distance constraint margin	m
$\boldsymbol{\delta}_{\text{robot}}$	Robot distance constraint margin	m
$\boldsymbol{\delta}_{\text{phase}}$	Phase constraint margin	$^{\circ}$
$\boldsymbol{\delta}_{L}$	Length scale parameter	-
$\boldsymbol{\delta}_{T}$	Time scale parameter	-

List of Acronyms

AIDA Asteroid Impact & Deflection Assessment. 8

D-H Denavit-Hartenberg. 20, 21

DART Double Asteroid Redirection Test. 8, 9

DART Lab Deep-space Astrodynamics Research & Technology Lab. 1, 8, 18

DLR Deutsche Forschungsanstalt für Luft- und Raumfahrt. 11, 13, 14, 35

DOF Degrees Of Freedom. 1, 9, 12–19, 22

ESA European Space Agency. 8, 11

FK Forward Kinematics. 18, 20, 21

FOV Field Of View. 3, 5, 6, 9, 10, 16

GNC Guidance, Navigation and Control. 1, 10

HIL Hardware-in-the-Loop. 1, 10, 11, 18

HMI Hydrargyrum Medium-arc Iodide. 12, 13

IK Inverse Kinematics. 18, 19, 22

IR Infrared. 14, 16

ISL Inter-Satellite Link. 8, 9

JPL Jet Propulsion Laboratory. 2

MTS Motion Tracking System. 16

NASA National Aeronautics and Space Administration. 2, 3, 5, 8

PoE product of exponentials. 20, 21

RAFFAELLO Robotic Arm Facility For Autonomous cubesat ExpLoration in cLose proximity Operations. 1, 10, 18

ROS Robot Operating System. 17

RSO Resident Space Object. 14

RvC Rendezvous and Capture. 11, 15

RvD rendezvous and docking. 11

SIL Software-in-the-Loop. 1

TRON Testbed for Robotic Optical Navigation. 13, 14, 35

TRON Testbed for Rendezvous and Optical Navigation. 14, 15, 35

V&V Verification and Validation. 1, 10

Acknowledgements

This work represents a conclusive chapter of my academic journey, and I would like to express my gratitude to everyone who has been part of it.

First and foremost, I want to thank my parents, who have supported every decision I have made, stood by me through every difficulty and given me the freedom to follow my path.

To my lifelong friends Paolo, Ale and Ale: thank you for always being there, even when university deadlines kept me from joining you.

Thank you to Elena, Pier, Pietro and Leo, my university friends with whom I have shared so many memories over the last years. A special thank you also goes to my climbing friends Gioele and Mark, who lightened my university weeks by training together.

A very special acknowledgment goes to Skyward Experimental Rocketry: joining this incredible team gave me so many opportunities to gain such valuable hands-on experience in the aerospace field and to work with many passionate and talented people.

Last, but not least, I would like to express my sincere gratitude to the DART Lab group. To my advisor and co-advisors, thank you for your guidance, your patience and your technical insight, especially during the last stressful days. This thesis would not have been possible without your constant support.