



POLITÉCNICA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES  
UNIVERSIDAD POLITÉCNICA DE MADRID

José Gutiérrez Abascal, 2. 28006 Madrid  
Tel.: 91 336 3060  
info.industriales@upm.es

[www.industriales.upm.es](http://www.industriales.upm.es)



Diego Ramírez Fuente

05 TRABAJO FIN DE MASTER

INDUSTRIALES

TRABAJO FIN DE MASTER

# DEVELOPMENT OF AUTONOMOUS MISSION COMPONENTS & PLANNING FOR THE EXECUTION OF THE ROBOTIC RADIATION SURVEY IN THE SPS

TRABAJO FIN DE MASTER  
PARA LA OBTENCIÓN DEL  
TÍTULO DE MASTER  
UNIVERSITARIO EN  
AUTOMÁTICA Y ROBÓTICA

FEBRERO 2026

**Diego Ramírez Fuente**

DIRECTOR DEL TRABAJO FIN DE MASTER:  
**Dr. Paloma de la Puente Yusty**  
**David Forkel**



POLITÉCNICA





## Acknowledgments

With this work, I bring to a close one of the most special chapters of my life. I would like to thank the BE-CEM-MRO section at CERN for giving me the opportunity to work with them during the development of my thesis. From day one, they made everything easy, made me feel at home even while being so far away from it, and were always there to help whenever I needed it, without hesitation and without expecting anything in return. In this context, I would especially like to thank my supervisor at CERN, David Forkel, for guiding me and supporting me throughout this entire journey, and my supervisor at UPM, Paloma de la Puente, who likewise helped me with her advice and encouragement during all this time.

I also do not want to forget to thank the wonderful friends I made during these months at CERN. They became my family throughout this beautiful period of my life, made my adaptation much faster, and, luckily, I know they will stay in my life forever as my friends.

In the same way, I want to thank the rest of my friends back in Spain, who, throughout my university years, were by my side at every moment and always supported me in everything I did.

Finally, and most importantly to me, I would like to thank my family, especially my parents, Jesús and Begoña, and my sister Irene, who have supported and loved me throughout my life, no matter what. I also want to remember my grandparents, Teresa, Carmen, and José, because I am convinced it would have meant so much to them to see what I am achieving. Thanks to them, and to having them by my side, I have had these wonderful opportunities, and I know I will continue to have them.



---

## Abstract

Gamma radiation surveys in the CERN Super Proton Synchrotron (SPS) tunnel are long-duration inspection missions carried out in a hazardous underground environment under strict operational constraints. Although autonomous navigation can support nominal traversal in unconstrained tunnel sections, end-to-end survey execution still requires reliable handling of mission-defining constrained phases, particularly safety-gate crossings and automatic docking for recharging. These phases traditionally demand frequent operator intervention, manual sequencing of mission modules, and unstructured recovery actions, which increases workload and reduces repeatability.

This thesis develops the mission-level autonomy required for the Measurement & Inspection Robot for Accelerators (MIRA) robot to execute full radiation surveys in the SPS tunnel with minimal operator intervention. The proposed solution is based on a supervised hierarchical Mission Manager implemented as a finite-state machine that orchestrates the survey workflow and coordinates lower-level navigation, perception, and safety behaviours. The Mission Manager structures the mission as a sequence of explicit states with clear transition conditions, enabling traceable execution and consistent procedure invocation. Dedicated autonomous procedures are integrated for safety-gate crossing and charging-station parking and docking, combining vision-based alignment and motion primitives to achieve precise positioning and safe trajectories in constrained geometries. Supervision and recovery are incorporated at system level to keep the operator in the loop for safety-relevant steps while reducing interaction to lightweight monitoring and discrete decision points.

Overall, the work contributes a structured and extensible framework that transforms the SPS robotic radiation survey from a predominantly teleoperated workflow into a supervised autonomous mission, improving operational consistency and supporting safer execution in accelerator tunnel environments.

**Keywords:** Mission manager, Super Proton Synchrotron (SPS), Measurement and Inspection Robot for Accelerators (MIRA), radiation survey, finite state machine (FSM), visual servoing, autonomous docking, gate crossing, error handling, recovery logic.

## UNESCO Codes

|             |                                |
|-------------|--------------------------------|
| <b>1203</b> | <b>Computer Science</b>        |
| 1203.17     | Computational Methods          |
| 1203.26     | Simulation                     |
| <b>3304</b> | <b>Technology of Computers</b> |
| 3304.91     | Robotics                       |
| <b>3311</b> | <b>Automation Technology</b>   |
| 3311.01     | Automation Technology          |
| 3311.02     | Control Engineering            |



---

## Resumen

Las inspecciones de radiación gamma en el túnel del Super Proton Synchrotron (SPS) de CERN son misiones de larga duración realizadas en un entorno subterráneo y peligroso, sujeto a restricciones operativas estrictas. Aunque la navegación autónoma puede asistir el desplazamiento nominal en tramos no restringidos del túnel, la ejecución completa de la inspección requiere gestionar de forma fiable fases críticas que definen la misión, en particular el cruce de compuertas de seguridad y el acoplamiento automático a la estación de carga. Tradicionalmente, estas fases requieren intervención frecuente del operador, secuenciación manual de módulos de misión y acciones de recuperación, lo que incrementa la carga de trabajo y reduce la repetibilidad.

Esta tesis desarrolla la autonomía a nivel de misión necesaria para que el robot de medida e inspección para aceleradores (MIRA) ejecute inspecciones completas de radiación en el túnel del SPS con una intervención mínima del operador. La solución propuesta se basa en un Gestor de Misión jerárquico y supervisado, implementado como una máquina de estados finitos, que orquesta el flujo de la inspección y coordina módulos de navegación, percepción y seguridad de bajo nivel. El Gestor de Misión estructura la misión como una secuencia de estados explícitos con condiciones de transición claras, permitiendo una ejecución trazable y una activación consistente de procedimientos. Se integran procedimientos autónomos específicos para el cruce de compuertas y para el aparcamiento y acoplamiento en la estación de carga, combinando alineamiento basado en visión y primitivas de movimiento para lograr un posicionamiento preciso y trayectorias seguras en geometrías restringidas. La supervisión y la recuperación se incorporan a nivel de sistema para mantener al operador en el bucle en pasos relevantes para la seguridad, reduciendo la interacción a monitorización ligera y a puntos discretos de decisión.

En conjunto, el trabajo aporta un marco estructurado y extensible que transforma la inspección robótica de radiación en el SPS de un flujo predominantemente teleoperado a una misión autónoma supervisada, mejorando la consistencia operativa y facilitando una ejecución más segura en entornos de túnel de acelerador.

**Palabras clave:** gestor de misión, Super Proton Synchrotron (SPS), Measurement and Inspection Robot for Accelerators (MIRA), inspección radiológica, máquina de estados finitos (FSM), servo control visual, acoplamiento autónomo, cruce de compuertas, manejo de errores, lógica de recuperación.

## Códigos UNESCO

|             |                                        |
|-------------|----------------------------------------|
| <b>1203</b> | <b>Informatica</b>                     |
| 1203.17     | Metodos computacionales                |
| 1203.26     | Simulacion                             |
| <b>3304</b> | <b>Tecnologia de computadores</b>      |
| 3304.91     | Robotica                               |
| <b>3311</b> | <b>Tecnologia de la automatizacion</b> |
| 3311.01     | Tecnologia de la automatizacion        |
| 3311.02     | Ingenieria de control                  |



# TABLE OF CONTENTS

|                                                                 |             |
|-----------------------------------------------------------------|-------------|
| <b>Acknowledgments</b>                                          | <b>II</b>   |
| <b>Abstract</b>                                                 | <b>IV</b>   |
| <b>Resumen</b>                                                  | <b>VI</b>   |
| <b>Index of Tables</b>                                          | <b>XII</b>  |
| <b>Index of Figures</b>                                         | <b>XV</b>   |
| <b>Index of Codes</b>                                           | <b>XVII</b> |
| <b>Abbreviations and Acronyms</b>                               | <b>XIX</b>  |
| <b>1 Introduction</b>                                           | <b>1</b>    |
| 1.1 Motivation . . . . .                                        | 1           |
| 1.2 SPS Facility and Operational Context . . . . .              | 2           |
| 1.3 Radiation Surveys in the SPS . . . . .                      | 2           |
| 1.4 MIRA Robotic Platform . . . . .                             | 3           |
| 1.5 Robotic Infrastructure in the SPS Tunnel . . . . .          | 4           |
| 1.6 From Teleoperation to Supervised Autonomy . . . . .         | 6           |
| 1.7 System Requirements and Assumptions . . . . .               | 8           |
| 1.8 Contributions to the SPS Radiation Survey Mission . . . . . | 9           |
| <b>2 State of the Art</b>                                       | <b>11</b>   |
| 2.1 Related Work and Technical Strategies . . . . .             | 11          |
| 2.2 Visual Servoing for Precision Robot Operations . . . . .    | 14          |
| 2.2.1 Control Strategies . . . . .                              | 15          |
| 2.2.2 ArUco Markers in Visual Servoing . . . . .                | 16          |
| <b>3 Existing Framework and Design Decisions</b>                | <b>19</b>   |
| 3.1 Mission Management Framework at CERN . . . . .              | 19          |

|          |                                                                     |           |
|----------|---------------------------------------------------------------------|-----------|
| 3.2      | Baseline SPS Survey Execution and Limitations . . . . .             | 20        |
| 3.3      | Finite-State Machine Implementation Using Boost SML . . . . .       | 21        |
| 3.4      | Visual Servoing with Fiducial Markers . . . . .                     | 22        |
| 3.5      | Design Rationale and Integration . . . . .                          | 22        |
| <b>4</b> | <b>Improved and New System Components</b>                           | <b>24</b> |
| 4.1      | Overview of the Visual Servoing Subsystem . . . . .                 | 24        |
| 4.1.1    | Marker Pose Detector . . . . .                                      | 24        |
| 4.1.2    | Kalman Filter for Marker Pose Tracking . . . . .                    | 26        |
| 4.1.3    | Target Visual Servoing Module . . . . .                             | 32        |
| 4.2      | Experimental Validation on the Kinova Gen2 Arm . . . . .            | 40        |
| 4.3      | Experimental Evaluation and Control Performance with MIRA . . . . . | 42        |
| 4.3.1    | Hybrid predictive–Kalman feedback. . . . .                          | 44        |
| 4.3.2    | Continuous Kalman-filtered feedback. . . . .                        | 45        |
| <b>5</b> | <b>Mission Manager for Autonomous Radiation Surveys</b>             | <b>48</b> |
| 5.1      | Mission Flow and Operational Logic . . . . .                        | 48        |
| 5.2      | Software Structure and Execution Flow . . . . .                     | 50        |
| 5.3      | Error Handling, Recovery and Operator Supervision . . . . .         | 53        |
| 5.4      | Developed Mission Components . . . . .                              | 56        |
| 5.4.1    | Gate Approaching and Gate Crossing . . . . .                        | 56        |
| 5.4.2    | Parking Procedure and Docking into the Charging Station . . . . .   | 58        |
| <b>6</b> | <b>Experimental Validation and Performance Results</b>              | <b>62</b> |
| 6.1      | Experimental Protocol and Metrics . . . . .                         | 62        |
| 6.2      | Gate Crossing Results . . . . .                                     | 63        |
| 6.3      | Parking and Docking Results . . . . .                               | 64        |
| 6.4      | Procedure Level and Mission Level Performance Comparison . . . . .  | 65        |
| 6.5      | Mission Level Timeline and Qualitative Interpretation . . . . .     | 67        |
| <b>7</b> | <b>Conclusions and Future Work</b>                                  | <b>69</b> |



|                   |                                                                         |           |
|-------------------|-------------------------------------------------------------------------|-----------|
| 7.1               | Conclusions . . . . .                                                   | 69        |
| 7.2               | Future Work . . . . .                                                   | 71        |
| <b>REFERENCES</b> |                                                                         | <b>76</b> |
| <b>ANNEXES</b>    |                                                                         | <b>77</b> |
|                   | Annex I - Sustainable Development Goals (SDGs) . . . . .                | 77        |
|                   | Annex II - Impact Evaluation and Ethical-Legal Considerations . . . . . | 79        |
|                   | Annex III - Temporal Planning and Execution Budget . . . . .            | 81        |
|                   | Annex IV - Video Demonstration . . . . .                                | 86        |



## Index of Tables

|     |                                                                                                                                                                                                                              |    |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1.1 | Comparison of SPS radiation survey execution in a predominantly teleoperated workflow and with the mission-level autonomy developed in this thesis. . . . .                                                                  | 10 |
| 2.1 | Integration of mission design strategies in related work. . . . .                                                                                                                                                            | 14 |
| 4.1 | Main configuration parameters of the <code>TargetVisualServoing</code> module and their mapping to the PBVS loop. . . . .                                                                                                    | 36 |
| 4.2 | Performance under raw <i>ArUco</i> feedback on the Kinova Gen2 arm (no Kalman filtering). $T_s$ is defined as the first time the 6D error enters the acceptance range.                                                       | 42 |
| 4.3 | Performance under hybrid predictive–Kalman feedback. Metrics reported as time to enter tolerance ( $T_s$ ) and average integrated squared error (ISE) across $X$ , $Y$ , and yaw. . . . .                                    | 45 |
| 4.4 | Performance of P and PD controllers under continuous Kalman-filtered visual feedback. Metrics reported as time to enter tolerance ( $T_s$ ) and integrated squared error (ISE), averaged across $X$ , $Y$ , and yaw. . . . . | 47 |
| 5.1 | Recovery logic used by the mission manager. . . . .                                                                                                                                                                          | 55 |
| 6.1 | Gate crossing timing ( $N = 50$ ). . . . .                                                                                                                                                                                   | 63 |
| 6.2 | PBVS residual error at convergence during gate approach ( $N = 50$ ). . . . .                                                                                                                                                | 63 |
| 6.3 | Parking procedure timing ( $N = 50$ ). . . . .                                                                                                                                                                               | 64 |
| 6.4 | PBVS residual error at convergence during parking alignment stages ( $N = 50$ ). .                                                                                                                                           | 64 |
| 6.5 | Procedure-level and mission-level performance comparison across operational modes.                                                                                                                                           | 66 |
| 7.1 | Execution budget breakdown. . . . .                                                                                                                                                                                          | 84 |
| 7.2 | Total execution budget summary. . . . .                                                                                                                                                                                      | 85 |

## Index of Figures

|     |                                                                                                                                                                                                                                                                                                    |    |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1.1 | Contextual and structural views of the SPS accelerator. . . . .                                                                                                                                                                                                                                    | 2  |
| 1.2 | Manual SPS ring survey using an electric vehicle and an odometry wheel (images courtesy of CERN Radiation Protection, RP). . . . .                                                                                                                                                                 | 3  |
| 1.3 | Overview of key MIRA hardware components used in SPS gamma radiation surveys. . . . .                                                                                                                                                                                                              | 4  |
| 1.4 | Safety gate modification for robot pass through. . . . .                                                                                                                                                                                                                                           | 5  |
| 1.5 | Charging infrastructure enabling underground docking and long-term operation. . . . .                                                                                                                                                                                                              | 5  |
| 1.6 | Schematic of the SPS tunnel layout for gamma radiation surveys [4]. . . . .                                                                                                                                                                                                                        | 6  |
| 1.7 | Autopiloted survey operation with two MIRA platforms running simultaneously. . . . .                                                                                                                                                                                                               | 7  |
| 2.1 | System architecture of the Hobbit robot [9]. . . . .                                                                                                                                                                                                                                               | 12 |
| 2.2 | Illustration of a three-layer control architecture where Behavior Trees (BTs) and Controlled Hybrid Dynamical Systems (CHDSs) operate in the middle layer, managing the switching between low-level controllers under high-level planning directives. Adapted from Marzinotto et al. [16]. . . . . | 13 |
| 2.3 | Comparison of eye-in-hand and eye-to-hand configurations. Illustrations generated by the author with AI support. . . . .                                                                                                                                                                           | 15 |
| 2.4 | Comparison of visual servoing strategies using fiducial markers: PBVS, IBVS, and hybrid. Illustration generated by the author with AI support. . . . .                                                                                                                                             | 16 |
| 2.5 | Functional block diagram of the vision-based control system. . . . .                                                                                                                                                                                                                               | 16 |
| 2.6 | Example of <i>ArUco</i> marker-based visual servoing for UAV precision landing [22]. . . . .                                                                                                                                                                                                       | 17 |
| 3.1 | External transition notation used to relate UML transitions to Boost SML transitionable syntax: source state, triggering event, optional guard, optional action, and destination state [29]. . . . .                                                                                               | 21 |
| 4.1 | Main constraints observed in <i>ArUco</i> based detections during MIRA operation in SPS. . . . .                                                                                                                                                                                                   | 25 |
| 4.2 | Comparison of Kalman filter configurations for marker pose estimation. . . . .                                                                                                                                                                                                                     | 29 |
| 4.3 | PBVS errors during a controlled marker occlusion. Orange: original. Blue: filtered (zeroed when lost). Red bands: marker lost. . . . .                                                                                                                                                             | 32 |
| 4.4 | UML diagram of the <code>TargetVisualServoing</code> module, showing its integration with detector, camera, and motion controller interfaces. . . . .                                                                                                                                              | 34 |
| 4.5 | Reference frames involved in the PBVS control loop for both setups. The transformations ${}^M\mathbf{T}_{Ta}$ and ${}^E\mathbf{T}_C$ are updated or calibrated depending on the robot configuration. . . . .                                                                                       | 38 |

|      |                                                                                                                                                                                                                                             |    |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 4.6  | Closed loop of the PBVS controller with internal Kalman estimation and Mahalanobis outlier rejection. In the hybrid predictive mode, the estimated state is propagated to the control timestamp, ensuring temporally consistent feedback. . | 39 |
| 4.7  | Experimental setup used for PBVS validation on the Kinova Gen2 arm. A fiducial marker is mounted on a fixed support, and an Intel RealSense camera is attached to the robot end-effector in an eye-in-hand configuration. . . . .           | 41 |
| 4.8  | Kinova Gen2 PBVS validation using raw <i>ArUco</i> pose feedback (no Kalman filtering). Illustrative traces shown for the selected PD configuration ( $K_p = 0.05$ , $K_d = 0.02$ ). . . . .                                                | 41 |
| 4.9  | Errors in $X$ , $Y$ , and yaw under the hybrid predictive-Kalman feedback for representative controllers. . . . .                                                                                                                           | 44 |
| 4.10 | Integrated squared error (ISE) for representative controllers under hybrid predictive-Kalman feedback. . . . .                                                                                                                              | 45 |
| 4.11 | Errors in $X$ , $Y$ , and yaw under continuous Kalman-filtered feedback for representative controllers. . . . .                                                                                                                             | 46 |
| 4.12 | Integrated squared error (ISE) for representative controllers under Kalman-filtered feedback. . . . .                                                                                                                                       | 46 |
| 5.1  | Mission flowchart representing the high-level operational logic of the autonomous survey procedure. . . . .                                                                                                                                 | 49 |
| 5.2  | UML class diagram of the <b>RPSurveySPS</b> mission manager and its core components.                                                                                                                                                        | 51 |
| 5.3  | Operator-in-the-loop architecture used in real and simulated execution. The FSM evaluates conditions autonomously, while progression is operator-gated through <b>Next</b> . . . . .                                                        | 53 |
| 5.4  | State machine diagram for the global error handling procedure. Recoverable failures transition from <b>WORKING</b> to <b>GLOBAL ERROR</b> , while safety-critical stops transition to <b>EMERGENCY</b> . . . . .                            | 54 |
| 5.5  | Gate marker placement. . . . .                                                                                                                                                                                                              | 57 |
| 5.6  | Gate crossing sequence. . . . .                                                                                                                                                                                                             | 57 |
| 5.7  | State machine diagram for the <b>GateCrossing</b> procedure. . . . .                                                                                                                                                                        | 58 |
| 5.8  | Charging station layout in the SPS tunnel. . . . .                                                                                                                                                                                          | 59 |
| 5.9  | <i>ArUco</i> marker placement on the charging station. . . . .                                                                                                                                                                              | 60 |
| 5.10 | Parking and docking sequence, front alignment, ramp staging, switch to rear perception, and reverse alignment plus final docking. . . . .                                                                                                   | 61 |
| 6.1  | Gate crossing execution time per step, mean and standard deviation ( $N = 50$ ). .                                                                                                                                                          | 63 |
| 6.2  | Gate crossing outcomes across trials ( $N = 50$ ). . . . .                                                                                                                                                                                  | 64 |
| 6.3  | Parking procedure execution time per step, mean and standard deviation ( $N = 50$ ). .                                                                                                                                                      | 65 |

|     |                                                                                                                                                  |    |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 6.4 | Parking procedure outcomes across trials ( $N = 50$ ).                                                                                           | 65 |
| 6.5 | Estimated mission timeline for MIRA1 and MIRA2.                                                                                                  | 68 |
| 7.1 | Train Inspection Monorail (TIM) platform for LHC tunnel operations, extensible<br>with additional payloads such as a robotic arm (TIM Arm) [34]. | 71 |
| 7.2 | Current operator GUI used for teleoperated gamma radiation surveys with MIRA.                                                                    | 72 |
| 7.3 | Example of visual-servoing-based screwing from the literature [24].                                                                              | 73 |
| 7.4 | Sustainable Development Goals (SDGs).                                                                                                            | 77 |



## Index of Codes

|     |                                                                                  |    |
|-----|----------------------------------------------------------------------------------|----|
| 3.1 | Illustrative Boost SML transition rules with event, guard, and action. . . . .   | 22 |
| 4.1 | Example JSON configuration for a <code>TargetVisualServoing</code> task. . . . . | 36 |





## Abbreviations and Acronyms

|              |                                                   |
|--------------|---------------------------------------------------|
| <b>CERN</b>  | European Organization for Nuclear Research        |
| <b>SPS</b>   | Super Proton Synchrotron                          |
| <b>LHC</b>   | Large Hadron Collider                             |
| <b>RP</b>    | Radiation Protection                              |
| <b>ALARA</b> | As Low As Reasonably Achievable                   |
| <b>MIRA</b>  | Measurement and Inspection Robot for Accelerators |
| <b>CRF</b>   | CERN Robotic Framework                            |
| <b>GUI</b>   | Graphical User Interface                          |
| <b>LiDAR</b> | Light Detection and Ranging                       |
| <b>IMU</b>   | Inertial Measurement Unit                         |
| <b>DoF</b>   | Degrees of Freedom                                |
| <b>IBVS</b>  | Image-Based Visual Servoing                       |
| <b>PBVS</b>  | Position-Based Visual Servoing                    |
| <b>SML</b>   | State Machine Language                            |
| <b>FSM</b>   | Finite State Machine                              |



## Chapter 1. Introduction

This chapter presents the context and motivation for mission-level autonomy enabling the MIRA robot to execute full radiation surveys in the SPS tunnel at CERN with minimal operator intervention. The SPS mission combines long-duration navigation with constrained procedures, particularly safety-gate crossings and autonomous docking for recharging. The chapter then describes the operational setting and the baseline system, and introduces the autonomy developments proposed in this thesis.

### 1.1 Motivation

The European Organization for Nuclear Research (CERN) operates the Super Proton Synchrotron (SPS), a key particle accelerator with a circumference of approximately 7 km [1]. The SPS is CERN's second-largest accelerator and constitutes the last injector stage in the accelerator chain that feeds particle beams to the Large Hadron Collider (LHC). In addition, it supplies beams to a broad experimental programme, including North Area experiments and associated test-beam facilities, as well as dedicated installations such as AWAKE and HiRadMat [2]. This operational role demands reliable inspection and maintenance procedures and imposes strict safety constraints on underground access.

The SPS tunnel is a harsh, semi-structured environment that includes ionising radiation, magnetic disturbances, repetitive geometry, and limited accessibility. These conditions make human interventions unsafe and inefficient, motivating robotic systems to enhance safety, maintainability, and machine availability while minimizing personnel exposure. In this context, Radiation Protection (RP) operations refer to activities performed in radiation-exposed or otherwise hazardous environments for radiological risk assessment, monitoring and control of radiation levels, and the execution of tasks that ensure personnel safety in accordance with CERN's ALARA (As Low As Reasonably Achievable) principle [3].

Robotic systems have already enabled remote inspection and RP-related operations at CERN. However, long tunnel missions still often rely on a predominantly teleoperated workflow, where operators must start and stop mission modules, manually manage transitions between phases, and intervene frequently to handle constrained manoeuvres and failures. In missions involving kilometers of repeating tunnel geometry, this manual orchestration increases workload, decreases repeatability, and makes execution dependent on operator time and judgment.

This work addresses these challenges by developing the mission-level autonomy required for the MIRA robot to execute SPS radiation surveys with minimal operator intervention. The focus is on a hierarchical mission manager and dedicated autonomous components for safety-gate crossing and automatic return-to-charge. Together, these elements enable supervised autonomy by reducing the operator role from step-by-step orchestration to lightweight monitoring while improving safety, repeatability, and consistency of survey execution.

## 1.2 SPS Facility and Operational Context

The SPS is a central element of CERN’s accelerator programme and supports both the LHC injector chain and a wide range of non-LHC users, including North Area experiments, test beams, and dedicated facilities [2]. The accelerator operates in cycles that include restricted-access periods and maintenance windows, during which radiation surveys are required to assess tunnel conditions and support safe planning of interventions.

From a robotics perspective, the SPS tunnel is a long semi-structured environment composed of repetitive arc sections interleaved with more complex zones around access and service areas, which combine long-duration traversal with discrete constrained events that require high precision and explicit safety procedures [4].

Figure 1.1 positions the SPS within CERN’s overall accelerator complex (Figure 1.1a) and provides a representative view of the SPS tunnel environment where the surveys take place (Figure 2.3b).

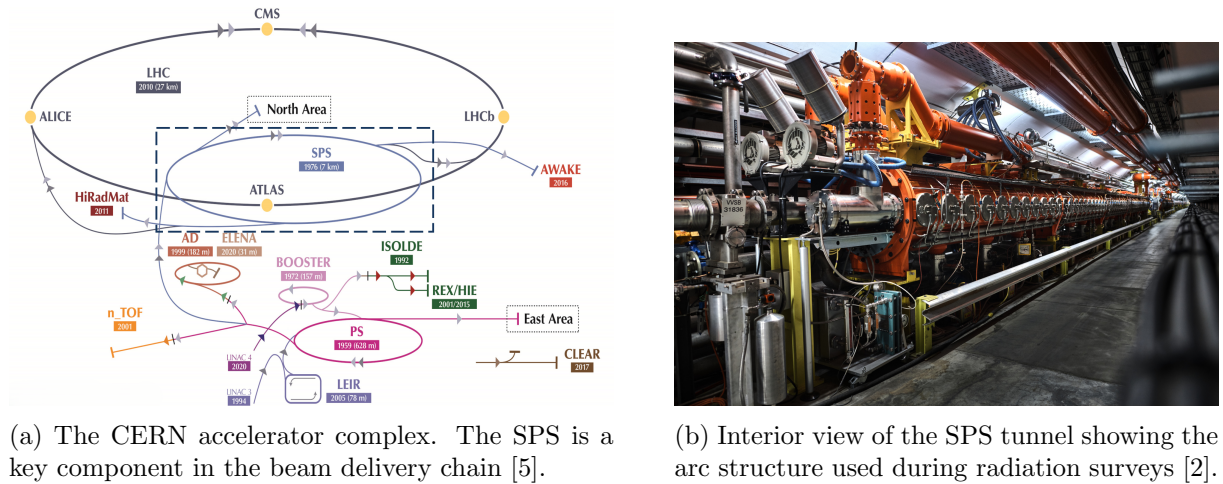


Figure 1.1: Contextual and structural views of the SPS accelerator.

## 1.3 Radiation Surveys in the SPS

Radiation surveys in the SPS are periodic measurement campaigns that quantify gamma radiation dose levels along the accelerator to support radiological assessment, job and dose planning, and operational diagnostics related to beam losses [4]. Survey requirements impose a strict measurement geometry to ensure comparability with historical datasets. During traversal, the radiation detector must, in particular, be kept at a specified clearance with respect to machine components. When necessary, additional measurements must be made in specific tunnel regions.

Before robotic deployment, these surveys were executed manually by CERN’s Radiation Protection group. A technician typically drove an electric vehicle along the SPS tunnel while continuously keeping track of dose-rate measurements, associating each reading with travelled distance obtained from an odometry wheel (Figure 1.2). This procedure is effective but requires tunnel occupancy and careful manual execution to preserve measurement consistency over long distances. In addition, repeated surveys may show variability due to changes in driving, alignment, stopping behaviour, and distance estimation.



(a) Survey setup with measurement trolley and odometry wheel for distance tracking and data association.



(b) Operator perspective during the survey using an electric vehicle and live acquisition monitoring.

Figure 1.2: Manual SPS ring survey using an electric vehicle and an odometry wheel (images courtesy of CERN Radiation Protection, RP).

Robotic execution reduces personnel exposure during routine monitoring and improves repeatability by enforcing consistent trajectories and measurement geometry. However, reliable robotic surveys in the SPS require that the mission can robustly handle constrained tunnel elements and safely transition between operational phases.

## 1.4 MIRA Robotic Platform

To meet the operational requirements for radiation protection in the SPS, two identical MIRA (Measurement and Inspection Robot for Accelerators) platforms have been permanently deployed in the SPS tunnel. The robots operate in opposite directions and perform independent surveys of complementary half-segments of the SPS ring, enabling full tunnel coverage with minimal infrastructure.

MIRA is a compact holonomic mobile robot developed at CERN with a modular design optimized for accelerator tunnel environments [6]. Its omnidirectional base uses four mecanum wheels, enabling lateral and diagonal motion that is essential for manoeuvring through narrow passages such as the safety-gate cut-outs. This locomotion, however, is prone to slippage on smooth surfaces, which motivates robust state estimation based on multi-sensor fusion and careful supervision during constrained manoeuvres.

The platform carries a 6 degrees of freedom robotic arm used to position the radiation detector according to survey measurement constraints. In the survey configuration, the gamma radiation sensor is mounted on the arm to maintain the required measurement geometry along the tunnel. During safety-gate crossings and docking, the arm configuration and the base trajectory must be actively managed to satisfy clearance and alignment constraints. A camera mounted on the end effector provides visual feedback for precision operations, such as visual servoing at gates and charging docks.

The on-board sensor suite supports both survey execution and remote supervision:

- **Proprioceptive sensors:** Wheel encoders and a VMU931 IMU for motion estimation and odometry.

- **Exteroceptive sensors:** RPLidar A3 2D laser scanner and multiple Axis F1035-E cameras for perception, monitoring, and remote supervision.
- **Mission sensors:** Gamma radiation detector and additional monitoring devices (e.g., audio alarms) used for safe operation.

All components are integrated within the CERN Robotic Framework (CRF), enabling modular communication, remote supervision, and mission execution through a Graphical User Interface (GUI). The framework supports TCP/IP communications, integration of mission modules, and real-time streaming of camera and sensor data for operator monitoring. In routine operation, an autopilot layer can support long-duration traversal in unconstrained arc sections by tracking the tunnel wall and machine elements using LiDAR- and vision-based perception, while dedicated behaviours handle constrained events such as gate traversal and docking.

The robot's main on-board computer used for mission execution is a LattePanda Sigma Small Form Factor (SFF) computer, providing the compute platform for running the mission stack and interfacing with sensors and CRF modules.



Figure 1.3: Overview of key MIRA hardware components used in SPS gamma radiation surveys.

### 1.5 Robotic Infrastructure in the SPS Tunnel

Executing autonomous radiation surveys in the SPS requires dedicated tunnel infrastructure that enables traversal across sector boundaries and long-term underground deployment with autonomous charging. Two infrastructure elements are mission-defining:

- **Safety gates:** The SPS ring is segmented by 19 fixed safety gates that separate tunnel sectors and control personnel access. To support robotic radiation surveys while preserving personnel access restrictions, these gates were modified by introducing a dedicated robot cut-out. The cut-out dimensions (20 cm by 40 cm) correspond to the maximum opening size allowed under the safety constraint of limiting human passage. As a consequence, this geometry became a primary requirement for the robotic platform and directly constrained the design envelope and traversal strategy of the MIRA base (Figure 1.4).

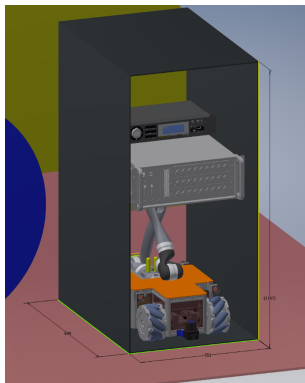


Because the clearance margins are tight in both lateral displacement and yaw, passing through a gate cannot be treated as nominal navigation. The robot must approach the gate, switch to a safe arm configuration, achieve precise alignment with the opening, execute a repeatable traversal motion, and then resume survey execution. These repeated constrained transitions motivate the dedicated autonomous gate-crossing procedures developed in this thesis.



Figure 1.4: Safety gate modification for robot pass through.

- **Charging stations:** Long-term underground operation also requires reliable recharging. In the SPS survey configuration used in this work, two charging stations are installed at BA3 and BA6 (Figure 1.6), where two identical MIRA robots are permanently deployed (Figure 1.5). Both robots operate in the same direction along the ring (clockwise) to ensure consistent survey execution. In nominal operation, the robot starting at BA3 surveys its assigned half of the SPS circumference and completes the mission by docking at BA6, while the second robot starting at BA6 surveys the complementary half and docks at BA3. This two-dock strategy simplifies logistics and enables full coverage while keeping the recharging infrastructure minimal.



(a) Conceptual design of the charging dock for underground deployment.



(b) Charging station installed in the SPS tunnel for automatic docking and recharging.

Figure 1.5: Charging infrastructure enabling underground docking and long-term operation.



A schematic overview of the SPS tunnel and survey context is shown in Figure 1.6, highlighting the sector divisions used for data segmentation and the distributed safety gates (yellow dashed lines) and charging locations that constrain mission execution.

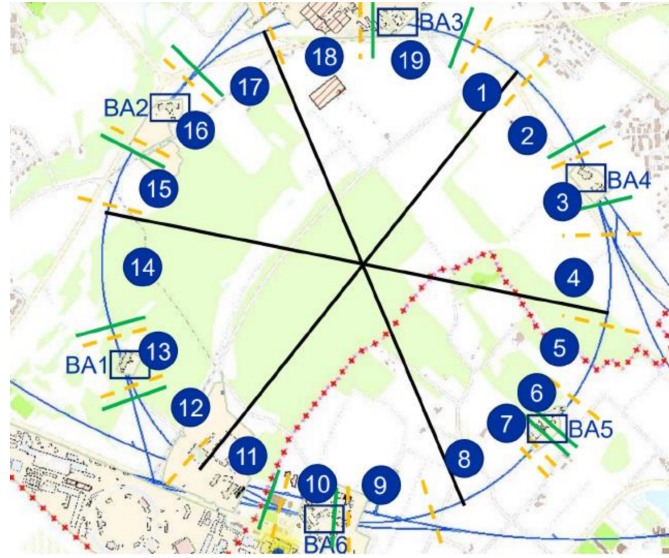


Figure 1.6: Schematic of the SPS tunnel layout for gamma radiation surveys [4].

## 1.6 From Teleoperation to Supervised Autonomy

Early SPS robotic surveys were executed in a predominantly teleoperated mode. The operator manually drove the mobile base, positioned the arm, aligned the robot by visual judgement, traversed constrained areas, and performed docking at the charging station. In practice, the full end to end survey depended on continuous human control and attention, which made execution slow, operator dependent, and difficult to reproduce consistently. Teleoperated execution in the SPS tunnel presents several recurring challenges:

- **Connectivity and delay:** Poor 4G coverage in parts of the tunnel can cause communication dropouts and delayed response times, which directly impacts manual control.
- **Time-consuming execution:** Continuous manual driving and arm positioning, together with step-by-step execution of critical manoeuvres, particularly gate traversal and docking, significantly increases overall mission duration.
- **Operator-dependent performance:** Alignment and positioning are performed by eye and depend on operator skill and timing, leading to variability between runs.
- **Inconsistency and recovery:** Without structured recovery logic, failures and unexpected events lead to improvised operator decisions, reducing repeatability and potentially affecting data integrity.
- **Safety compliance risk:** Reliance on continuous operator input increases the probability of human error, which is undesirable under ALARA constraints.
- **Stress and fatigue:** Long missions require constant attention and timely interventions, increasing workload and fatigue.

To mitigate these limitations, an autonomy baseline has been progressively introduced for SPS surveys. This baseline provides lower-level motion control and perception pipelines, together with an assisted-driving or autopilot mode that supports nominal survey execution in unconstrained tunnel sections. In particular, the autopilot enables autonomous traversal of the SPS arc sections, corresponding to approximately 90% of the tunnel length, while maintaining a continuous distance to the tunnel wall and beam components at a nominal speed of about 1.5 m/s. Beam-component recognition allows consistent assignment of measurement data to tunnel elements, and onboard obstacle detection triggers automatic safety stops. In addition, the autopilot can bridge short intervals of poor 4G coverage by enabling unsupervised operation for limited time windows [7].



(a) Autopilot test in the SPS tunnel, maintaining a constant distance to beam components during nominal arc traversal.



(b) Robotic Radiation Survey executed by two operators simultaneously in the CERN Control Centre.

Figure 1.7: Autopiloted survey operation with two MIRA platforms running simultaneously.

In routine operation, more than 136 km have been covered using this autopilot baseline across MIRA1 and MIRA2, reducing total operation time by more than one hour. Despite this level of autonomy, the autopilot is explicitly limited to nominal tunnel traversal and does not address constrained areas of the mission.

Despite the availability of an autopilot baseline for nominal survey execution, two mission phases remain major operational bottlenecks and motivate the introduction of dedicated mission-level autonomy.

The first bottleneck is **safety-gate traversal**. Safety gates impose tight clearances and require the robot to execute a structured sequence of actions, including controlled approach, precise alignment, safe arm configuration, traversal through the gate cut-out, and recovery to nominal survey operation. These manoeuvres cannot be reliably handled by the autopilot, which is designed for unconstrained motion and lacks the precision and sequencing required in such environments.

The second bottleneck is **automatic return-to-charge**. A complete survey mission must terminate with a safe and reliable docking procedure at the charging station. This phase requires accurate alignment, controlled motion under tight geometric constraints, and robust handling of alignment or docking failures. As with gate traversal, these requirements exceed the scope of the existing autopilot functionality.

This work therefore focuses on mission-level autonomy rather than on developing new navigation or autopilot capabilities. The contribution lies in the supervision, sequencing, and recovery logic required to robustly execute these critical mission phases. A hierarchical mission manager formalizes the survey as a structured sequence of states, coordinating existing lower-level navigation and perception modules with safety behaviours and dedicated motion primitives.

Visual servoing based on fiducial markers provides high-precision alignment at safety gates and charging docks, while mission-level recovery actions and safe fallback states reduce the need for continuous operator intervention.

### 1.7 System Requirements and Assumptions

The autonomous mission manager is designed under a set of requirements that define both its software architecture and the operational assumptions under which it functions within the SPS environment.

#### Architectural Requirements

The architecture must guarantee modularity, robustness, and interoperability with the CRF. The main requirements are:

- **Modularity:** Mission components (survey execution, gate crossing, return-to-charge, safety supervision) must be implemented as independent modules following CRF communication standards, enabling reuse and maintainability.
- **Hierarchical control:** Mission coordination shall be managed through a hierarchical finite-state machine capable of handling task transitions, synchronization, and recovery.
- **Fault tolerance:** Recovery actions must handle common failure modes such as alignment failure at gates or docks, localization dropouts, or communication interruptions, including safe stop and operator fallback states.
- **Interoperability:** All modules must interface with CRF middleware using standardized messaging and configuration formats.
- **Real-time execution of critical behaviours:** Precision phases (servoing and motion primitives) must run with deterministic timing and include safety supervision.
- **Configurability:** Mission parameters (gate list, traversal direction, docking target, survey constraints) must be configurable without code changes.

#### Operational Assumptions

The mission logic is based on three assumptions.

First, the survey must guarantee complete tunnel coverage under strict measurement conditions. The robot must maintain the required clearance and execute continuous acquisition with consistent geometry to ensure comparability with historical survey datasets.

Second, the system assumes prior knowledge of fixed tunnel infrastructure, including the location of the 19 safety gates and the two charging stations at BA3 and BA6. The mission manager uses this knowledge to trigger dedicated gate-crossing behaviours and to plan automatic return-to-charge at the appropriate docking station.

Third, the mission assumes that gate traversal and docking require dedicated precision routines beyond generic navigation. Visual Servoing provides reliable local references to achieve repeatable alignment and safe trajectories under tight clearances.

## 1.8 Contributions to the SPS Radiation Survey Mission

This work presents the mission-level autonomy required for the MIRA robot to execute SPS radiation surveys with minimal operator intervention in the constrained phases of the mission. The contributions focus on a hierarchical mission manager and dedicated autonomous components for safety-gate traversal and end-of-mission docking, building on existing lower-level navigation and perception capabilities.

The main contributions are:

- **Autonomous safety-gate crossing procedures:** Development of dedicated behaviours for controlled approach, precise alignment, safe arm configuration, traversal through the gate cut-out, and recovery to nominal survey operation, using visual servoing and motion primitives to improve repeatability and safety.
- **Automatic return-to-charge:** Development of an autonomous end-of-mission procedure that transitions from survey execution to docking, performs alignment and docking at the charging station, and includes recovery actions and safe fallback states.
- **Hierarchical mission manager:** Implementation of a mission-level hierarchical FSM that coordinates high-level tasks and orchestrates lower-level navigation, perception, and safety behaviours through event-driven transitions and supervision.
- **Integration towards supervised autonomy:** Integration of the above mission-level components in CRF to reduce operator involvement from continuous teleoperation in constrained phases to lightweight monitoring, improving safety compliance, repeatability, and operational efficiency.

| Aspect                        | Teleoperated surveys                                                                                                                                            | With this work                                                                                                      |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| Gate crossing                 | Operator manually drives and aligns the platform, positions the arm, and traverses the gate by continuous teleoperation, typically relying on visual judgement. | Autonomous gate crossing procedure using servoing and motion primitives.                                            |
| Arm positioning and alignment | Operator manually moves the arm to the desired pose and performs fine alignment by eye, with continuous corrective inputs.                                      | Automated arm positioning within the mission logic, with alignment handled by closed loop procedures when required. |
| Mission sequencing            | Operator executes the full procedure step by step, selecting actions, starting and stopping modules, and handling transitions manually.                         | Hierarchical mission manager orchestrates tasks through event driven FSM states.                                    |
| Return-to-charge / docking    | Operator manually navigates to the station and performs docking, including recovery from misalignment or failed attempts.                                       | Autonomous return-to-charge with mission-level recovery and safe fallback.                                          |
| Operator role                 | High workload and continuous attention, with manual control of base and arm for most actions and frequent intervention.                                         | Lightweight monitoring, intervention only on exceptional failures.                                                  |
| Repeatability                 | Sensitive to operator skill, timing, and subjective alignment, leading to variability between runs.                                                             | Standardized execution improves repeatability and consistency.                                                      |

Table 1.1: Comparison of SPS radiation survey execution in a predominantly teleoperated workflow and with the mission-level autonomy developed in this thesis.

## Chapter 2. State of the Art

Mission management has become a central topic in robotics, spanning industrial, service, and field domains. Diverse approaches, ranging from Finite-State Machines (FSMs) and Behavior Trees (BTs) to Petri Nets and hybrid planners, have been proposed to balance autonomy, safety, and adaptability. This chapter reviews representative mission management frameworks and their relevance to long-term autonomous operations. Emphasis is placed on hierarchical architectures, event-driven transitions, and recovery strategies, as these principles directly inform the design approach followed in subsequent chapters.

While CERN has developed its own robotic framework for radiation environments, its structure and design rationale are presented separately in Chapter 3. The present chapter focuses instead on general research advances and robotic systems that have influenced the development of autonomous mission management architectures.

### 2.1 Related Work and Technical Strategies

Autonomous mobile manipulators designed for inspection and intervention tasks in complex environments have been widely studied in recent years. Their mission managers typically employ FSM structures, event-driven paradigms, or hybrid architectures combining multiple control strategies. This subsection presents relevant contributions that form the conceptual basis for mission-level coordination and autonomy in robotic systems.

#### **Hierarchical Finite-State Machines (HFSM):**

Zhou et al. [8] proposed a hierarchical FSM architecture for autonomous mobile manipulators performing manufacturing and inspection tasks. Their system decomposes missions into high-level macro-states (e.g., **INIT**, **INTERACTION**, **PLANNING**, **IMPLEMENT**) and embeds low-level FSMs within implementation periods for execution primitives such as manipulation or navigation. This layered structure improves modularity, task clarity, and robustness by isolating decision-making from physical execution. Similar hierarchical decompositions are now widely used to manage complex robotic tasks, allowing concurrent control of navigation, perception, and recovery routines within a unified mission framework.

A comparable hierarchical approach was adopted in assistive robotics, notably in the EU-funded **Hobbit** project [9, 10, 11]. Hobbit relied on **ROS SMACH** to implement complex hierarchical state machines coordinated by a supervisory module that integrated multimodal inputs (speech, touchscreen, gestures, and emergency buttons). The mission manager orchestrated multiple modules such as **GoTo**, **Recharge**, **FindObject**, and **PickUp**, each encapsulating perception, planning, and actuation routines. This design demonstrated that FSM-based architectures can handle multimodal interaction and safety-critical decision-making while preserving modularity and robustness principles equally relevant for industrial, service, and field robotics.

Similar hierarchical mission architectures have also been explored in aerial robotics. The *Aerostack* framework [12] apply layered FSM and task-based structures to coordinate autonomous UAV operations. This framework emphasizes modularity, knowledge-driven behavior representation, and hybrid control strategies that combine deterministic state execution with semantic reasoning—further illustrating the versatility of hierarchical FSM approaches across robotic domains.

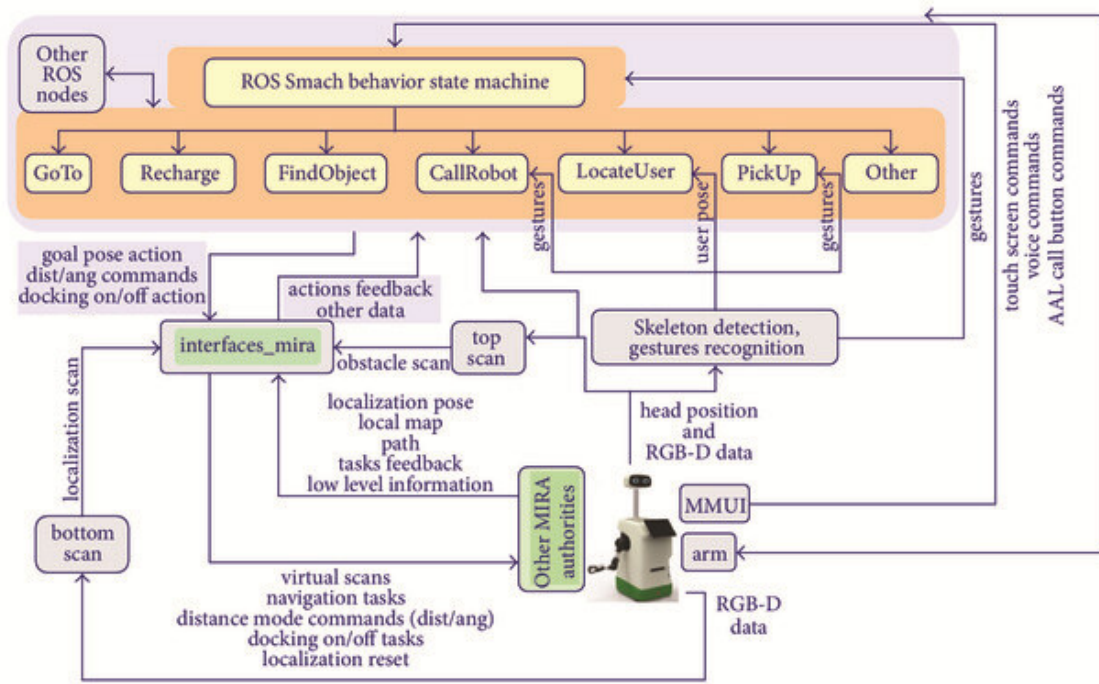


Figure 2.1: System architecture of the Hobbit robot [9].

### Event-Driven Control:

Janisz et al. [13] highlighted the advantages of event-driven robotic control, where asynchronous sensor or system events trigger transitions between states. This approach has been applied to various domains, such as reactive walking in humanoid robots [14], where environment feedback governs gait adaptation. Event-based architectures enhance responsiveness and simplify synchronization between perception and action layers, making them suitable for systems operating under dynamic or partially structured conditions.

A similar hierarchical approach was adopted in assistive robotics projects such as the EU-funded Hobbit robot [9], where complex FSMs coordinated multimodal interaction, navigation, and user assistance routines, demonstrating the scalability of hierarchical state-based control beyond industrial contexts.

### Constraint Monitoring and Reactive Safety:

Leutenegger et al. [15] demonstrated the use of real-time constraint monitoring, such as distance, velocity, and proximity limits to enable immediate safety reactions. Continuous monitoring of mission constraints ensures that the robot can detect violations and trigger appropriate recovery or emergency procedures. This principle underpins many modern mission frameworks, which implement safety checks as parallel processes running independently of the main control loop to guarantee timely intervention during anomalies.



### Behavior Trees for Robust Recovery:

Marzinotto et al. [16] introduced a unified framework for Behavior Trees (BTs) that addresses several limitations of traditional FSMs. Their formalization of success (S), failure (F), and running (R) states, along with mechanisms such as memory nodes (Node\*) and priority-based selectors, provides structured yet flexible recovery behavior. BTs allow systems to manage failure handling hierarchically and to retry failed actions in a controlled manner before escalating to operator intervention. This approach offers higher adaptability than purely deterministic FSMs while maintaining a clear execution logic.

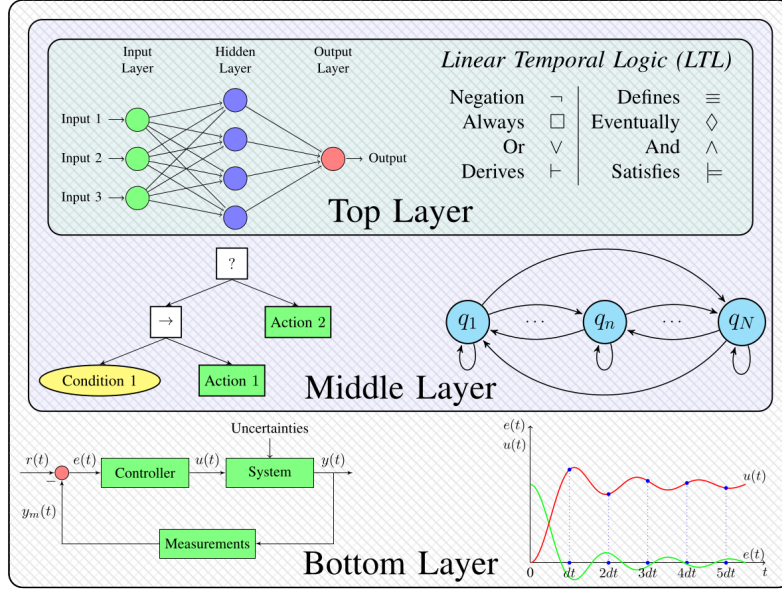


Figure 2.2: Illustration of a three-layer control architecture where Behavior Trees (BTs) and Controlled Hybrid Dynamical Systems (CHDSs) operate in the middle layer, managing the switching between low-level controllers under high-level planning directives. Adapted from Marzinotto et al. [16].

### Mission-Level Autonomy and Distributed Control (NeBula):

The NeBula architecture [17] introduced a modular, risk-aware mission autonomy system designed for multi-robot coordination in GPS-denied and communication-degraded environments. Its hierarchical planning structure comprises a Mission Executive, modular behaviors, a Mission Watchdog, and a Copilot assistant for system health monitoring and human interaction. Although originally conceived for distributed exploration missions, its principles—such as modular sequencing, layered autonomy, and operator offloading—are broadly applicable to inspection and survey robots. While NeBula employs probabilistic reasoning and behavior retasking based on uncertainty, FSM- or BT-based approaches typically emphasize deterministic transitions and safety guarantees, offering complementary paradigms for mission-level control.



### Comparative Perspective:

All these approaches provide different contributions to robot autonomy at the mission level. Zhou’s hierarchical FSM targets decomposition that introduces additional modularity and transparency into the process principles that we employ in addressing macro- and microbehavior in SPS. NeBula displays robust coordination under uncertain conditions. We mirror their implementation of modularity while maintaining deterministic transitions. Marzinotto’s BT approach provides formal ways of enhancing our error recovery without compromising the mission’s deterministic FSM core. The Hobbit project highlights the benefits of explicitly modularized task encapsulation with ROS SMACH and safe task interruption, demonstrating how socially assistive contexts also require robust supervisory logic; these principles map directly to SPS sub-machines like **GateCrossing** and **PreChargePositioning**. The combination creates a system specially designed to suit the SPS tunnel’s operational requirements, combining verifiable state transitions with adaptive recovery strategies.

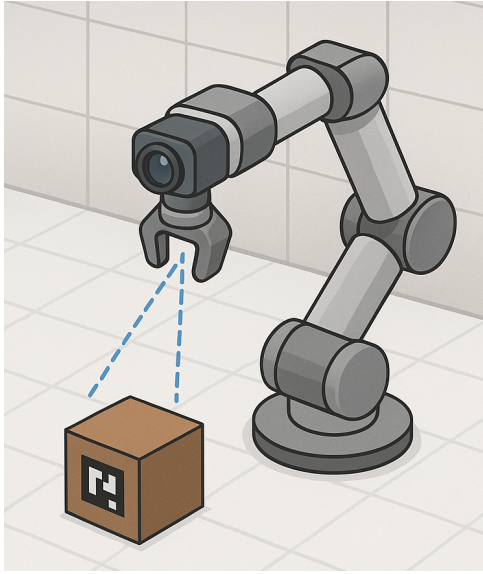
| Approach                        | Applied to Mission                                                                                                                  | Reference                      |
|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|
| Hierarchical FSM                | Macro states such as <b>INIT</b> and <b>NAVIGATION</b> , with nested states for <b>ERROR_HANDLING</b> .                             | Zhou et al. (2017) [8]         |
| Event-driven Control            | Sensor events, for example <b>gate_detected</b> and <b>low_battery</b> , trigger state transitions.                                 | Janisz et al. (2014) [13]      |
| Constraint Monitoring           | Safety enforced through constraints such as beam distance and radiation thresholds.                                                 | Leutenegger et al. (2014) [15] |
| Behavior Trees                  | BT-inspired fallback logic integrated within <b>ERROR_HANDLING</b> .                                                                | Marzinotto et al. (2014) [16]  |
| Mission Architecture            | Modular execution with operator offloading when required.                                                                           | NeBula [17]                    |
| SMACH-based Supervisory Control | Modular task orchestration and safe interruption, for example <b>PickUp</b> and <b>Recharge</b> , transferable to SPS sub-machines. | Hobbit Project (2016) [9]      |

Table 2.1: Integration of mission design strategies in related work.

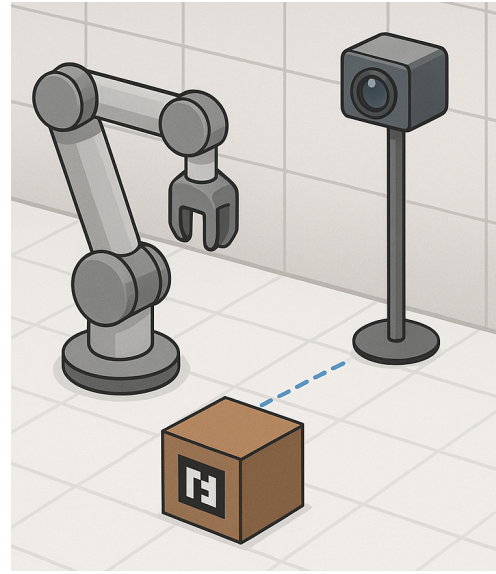
## 2.2 Visual Servoing for Precision Robot Operations

Visual Servoing (VS) is a closed-loop control technique that uses visual sensor feedback, typically from one or more cameras, to command the motion of a robotic system in real time. The control objective is defined relative to visual features of a target (image points, contours, or fiducial markers), and the robot adjusts its motion to minimize the error between the current and desired visual observations.

VS configurations are traditionally categorized into *eye-in-hand* (camera on the robot’s end-effector) and *eye-to-hand* (camera fixed in the environment), each offering different advantages. Eye-in-hand setups allow for precise manipulation feedback and dynamic re-targeting, while eye-to-hand setups provide a global view of the scene, ideal for mobile platforms and cooperative manipulation.



(a) Eye-in-hand



(b) Eye-to-hand

Figure 2.3: Comparison of eye-in-hand and eye-to-hand configurations. Illustrations generated by the author with AI support.

### 2.2.1 Control Strategies

Two main approaches dominate the field of visual servoing: **Image-Based Visual Servoing (IBVS)** and **Position-Based Visual Servoing (PBVS)** [18].

In **PBVS**, the system minimizes the Cartesian error in Euclidean space and predicts the target's whole three-dimensional pose in relation to the camera. This approach produces smooth and predictable trajectories in task space, which is advantageous for applications requiring precise positioning, such as assembly, docking, or gate crossing. However, precise camera calibration and a precise geometric representation of the target are essential for PBVS. Any deviation in these parameters can introduce instability or convergence errors.

In contrast, **IBVS** computes control instructions that move the image features toward their desired positions in the picture plane and defines the control objective only in terms of image features, such as pixel coordinates or centroids. This removes the need for explicit pose estimation, improving robustness against calibration errors or model inaccuracies. IBVS automatically maintains target visibility during motion since the control is established in the picture domain, which is a big benefit in situations that are cluttered or likely to occlusion. However, IBVS can generate 3D pathways that are not intuitive or ideal, and its performance may deteriorate when there are significant rotations or out-of-plane displacements, where depth coupling becomes critical.

To address the limitations of both approaches, several hybrid or 2.5D visual servoing strategies have been proposed, combining 2D image features with partial 3D information to balance stability and accuracy [19]. While using the geometric structure of PBVS to guarantee smoother motion and improved convergence, these techniques aim to maintain the visual robustness of IBVS.

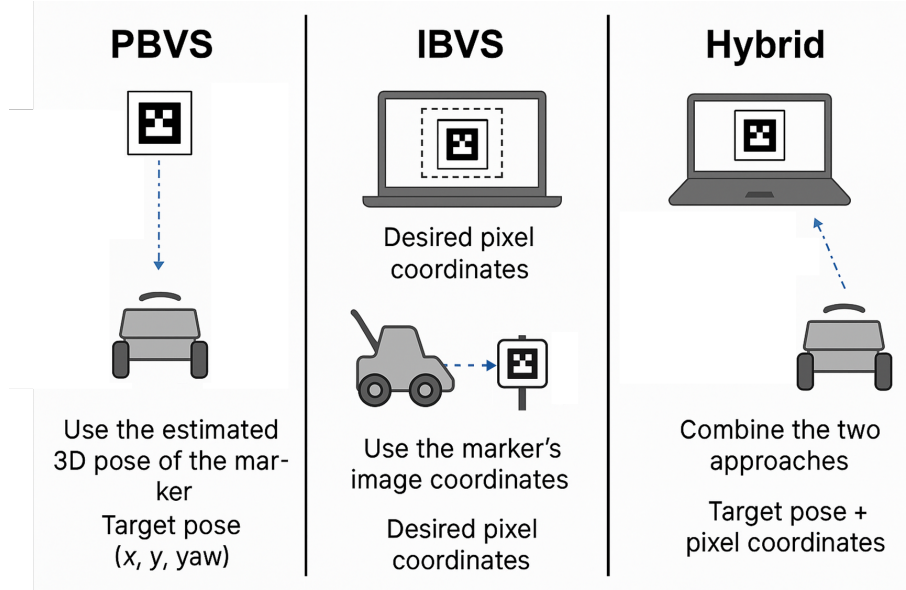


Figure 2.4: Comparison of visual servoing strategies using fiducial markers: PBVS, IBVS, and hybrid. Illustration generated by the author with AI support.

To facilitate reliable 6-DoF pose estimation and precise feedback control, modern PBVS implementations often rely on fiducial markers such as *ArUco*.

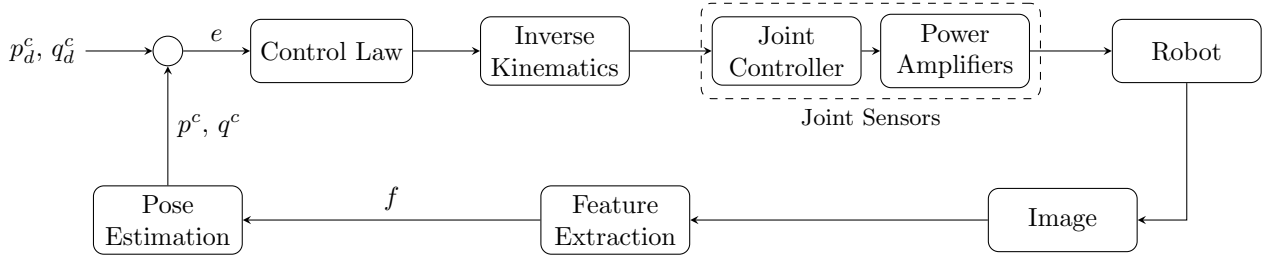


Figure 2.5: Functional block diagram of the vision-based control system.

## 2.2.2 ArUco Markers in Visual Servoing

Fiducial markers such as *ArUco* [20, 21] have become a key component of modern visual servoing, particularly in PBVS. Their well-defined geometry and efficient detection algorithms enable precise 6-DoF pose estimation from a single camera, providing reliable feedback for alignment, docking, and manipulation tasks. With minimum computational cost, *ArUco* markers may be easily integrated into real-time control pipelines thanks to their inherent support by *OpenCV* [21].

The *ArUco* system, first presented by Garrido-Jurado et al. [20], offered a way to automatically generate and detect highly reliable fiducial markers, even in the presence of partial occlusion. Its design approach minimizes recognition mistakes due to perspective distortion or noise by optimizing the inter-marker Hamming distance. The detection pipeline provides consistent 6-DoF tracking in real time by combining PnP-based pose estimation, adaptive thresholding, and sub-pixel corner refining. The system also introduced the concept of *marker boards*, in which several tags are arranged in a known geometric configuration. By combining the information from multiple markers, the system achieves higher pose accuracy and greater robustness against occlusion or partial detection. This feature has made ArUco especially useful in robotics and

augmented reality, where reliable pose estimation is essential for tasks such as grasping, navigation, or camera calibration.

In typical PBVS implementations, *ArUco* markers serve as ideal pose references. The control loop generally involves detecting the marker and its ID, estimating the 6D pose via PnP, filtering the result (e.g., using a Kalman or exponential smoother), and feeding the pose error into a velocity or position controller.

The versatility of *ArUco*-based visual servoing has led to its adoption across multiple fields:

- **Drone Landing and Navigation:** Wynn and McLain [22] demonstrated precision quadrotor landing on a pad using concentric *ArUco* markers, achieving robust detection under both daylight and nighttime conditions. Their system combined IBVS for image centering with pose feedback for altitude refinement.
- **Assembly and Manipulation:** Fernandez-Ayala et al. [23] guided a mobile manipulator to grasp and assemble blocks using *ArUco*-based eye-in-hand tracking, integrating local servoing with global planning.
- **Conveyor Tracking:** González-Huarte and Ibarguren [24] developed a real-time control architecture where a mobile manipulator followed a moving part by tracking its *ArUco* tag during a screw-fastening task.

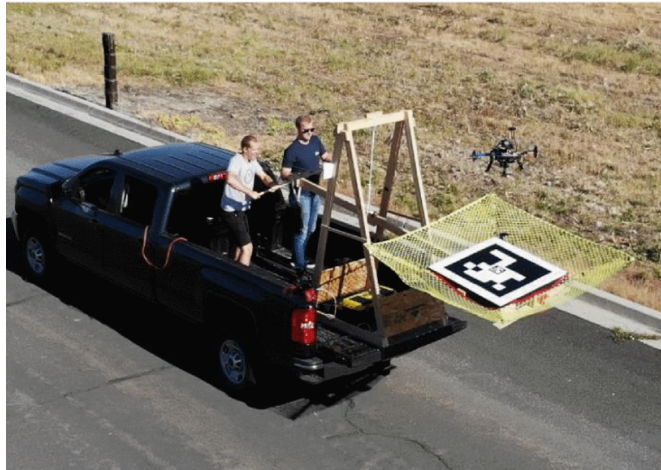


Figure 2.6: Example of *ArUco* marker-based visual servoing for UAV precision landing [22].

While *ArUco*-based visual servoing offers low-latency, high-accuracy pose feedback, it also presents constraints:

- **Lighting sensitivity:** Contrast and exposure have an impact on detection quality. Exposure control and picture preparation (such as histogram equalization) are frequently combined to guarantee accurate results.
- **Pose jitter:** Pose noise in lateral axes and yaw requires temporal filtering.
- **Field-of-view limitations:** The minimum operating distances and allowed approach angles are limited because the *ArUco* marker needs to be in the camera's field of view during the whole movement. Some systems compensate by dynamically adjusting the zoom or tilt of the camera.

Despite these challenges, marker-based PBVS is still one of the most useful strategies for field robotics. By embedding *ArUco*-based servoing modules into FSM or BT-based mission frameworks, tasks such as gate traversal, charging alignment, and grasp correction become more reliable and accurate.

## Chapter 3. Existing Framework and Design Decisions

This chapter describes the existing robotic framework developed at CERN and the design decisions adopted to extend it for the SPS radiation survey mission. The presented architecture builds upon the CRF, which provides the modular basis for mission control and low-level actuation. On top of this framework, additional mission-specific modules were integrated, including a visual-based precision manoeuvring system and a dedicated Mission Manager. These extensions were motivated by the limitations observed during teleoperated SPS surveys, where nominal navigation support existed but constrained procedures and mission-level recovery still required continuous manual intervention.

### 3.1 Mission Management Framework at CERN

The CERN Robotic Framework is a modular software system developed to support robotic operations in accelerator environments [25]. It defines the core components required for robot control and supervision, serving as a foundation on which mission-specific modules can be integrated. Mission logic is typically expressed using FSMs implemented with Boost State Machine Language (SML) [26]. The framework supports hierarchical and orthogonal FSMs for concurrent processes such as emergency monitoring, while maintaining deterministic execution and low runtime overhead with tight integration to supervisory GUIs.

A typical mission implementation in the CRF includes:

- A generic `IMissionManager` interface defining standard mission commands (`start`, `pause`, `stop`, `resume`, `emergency`).
- A Boost SML state machine defining *events*, *guards* and *actions* to represent mission logic.
- TCP/IP communication modules for exchanging commands and telemetry with external GUIs, enabling human-in-the-loop operation when required.

The CRF is used across multiple CERN facilities, including the Beam Loss Monitor (BLM) inspection missions [27]. Its modularity allows mission logic to be coupled with low-level motion controllers through standardized interfaces. In particular, `IMotionController` defines common abstractions for trajectory execution and task-space control, while the `PathFollower` module interpolates commanded paths, enforces dynamic constraints, and ensures smooth waypoint transitions. These modules abstract robot-specific implementations and enable a consistent control layer across different robotic platforms.

#### Baseline capabilities provided by the CRF.

In summary, the CRF already provided the following capabilities used in this work:

- Modular mission infrastructure with a generic `IMissionManager` interface and GUI connectivity.
- A Boost SML-based approach for implementing hierarchical and orthogonal FSMs.

- Standardized motion-control abstractions (`IMotionController`) and trajectory execution utilities (e.g., `PathFollower`).
- Middleware conventions and message formats to integrate perception, navigation, and control modules.

## 3.2 Baseline SPS Survey Execution and Limitations

Before the work presented in this thesis, SPS radiation surveys relied on teleoperated missions built on top of the CRF. The CRF already provided generic mission management interfaces, motion control modules, and communication with supervisory GUIs. An auxiliary autopilot was available to stabilise motion along the tunnel by maintaining alignment with the geometry through reactive LiDAR-based corrections. However, it was limited to short-range trajectory assistance, had to be manually activated by the operator, and was restricted to nominal, unconstrained traversal.

In practice, these baseline capabilities improved efficiency during routine tunnel motion but did not address the main operational bottlenecks. Constrained sections such as safety-gate traversal were not handled by the autopilot, and critical tasks such as gate crossing and docking at the charging station remained fully teleoperated due to the absence of closed-loop alignment methods. In addition, the lack of mission-level sequencing and structured recovery meant that long surveys required frequent manual interventions, especially when communication interruptions, localisation degradation, or procedure failures occurred.

These limitations motivated the introduction of mission-level autonomy for SPS surveys. A dedicated Mission Manager was required to coordinate navigation, radiation measurement, safety-gate traversal, and charging operations under strict safety and positioning constraints, while maintaining operator supervision for safety-critical decisions. The design and implementation of this mission-specific controller are presented in Chapter 5.

### Contributions introduced in this work.

On top of the CRF baseline, this thesis introduces mission-specific modules and logic to enable long-duration supervised autonomy for SPS radiation surveys:

- A dedicated Mission Manager that formalises the SPS survey procedure and sequences navigation, measurement, safety-gate traversal, and return-to-charge.
- A hierarchical FSM structure with mission-level recovery actions and safe fallback states tailored to SPS operational constraints.
- Visual servoing based on fiducial markers for high-precision alignment at safety gates and charging stations, integrated with CRF motion-control abstractions.

In the remainder of this chapter, the focus remains on the CRF mechanisms and design choices that underpin these extensions, while Chapter 5 details the mission-specific logic and hierarchical FSM structure.



### 3.3 Finite-State Machine Implementation Using Boost SML

Within the CRF, mission logic is typically implemented as a hierarchical finite-state machine using Boost SML, a modern C++ library for static state machines [28]. Mission behaviour is expressed through transition-table rules that combine a triggering event with optional guards and actions. This encoding makes transition conditions and side effects explicit, which is particularly suitable for safety-critical robotic missions where mission logic must be reviewable and deterministic.

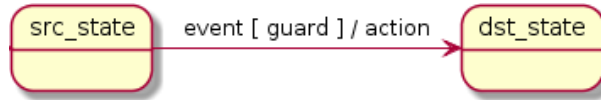


Figure 3.1: External transition notation used to relate UML transitions to Boost SML transition-table syntax: source state, triggering event, optional guard, optional action, and destination state [29].

Figure 3.1 provides a compact view of how Boost SML encodes the core elements of mission logic, namely *events*, *guards*, and *actions*, into a transition-table rule [29]. In this paradigm, events capture discrete triggers, guards enforce preconditions, and actions interface with the underlying CRF modules to execute mission steps.

- **Events:** Triggers that initiate state transitions are represented by events. Operator instructions, internal mission-flow signals such as `next`, and alerts produced by perception, navigation, and system-monitoring modules are typical sources. Events provide the mechanism by which mission-relevant information reaches the mission logic.
- **Guards:** Boolean conditions evaluated during event processing are represented by guards. Their role is to determine whether a transition is permitted under current mission conditions. Guards ensure that the system only progresses when the required conditions are satisfied, including constraints related to measurement points, gate detection, navigation progress, localisation quality, and battery level.
- **Actions:** Operations executed during a transition are represented by actions. In the CRF, actions typically communicate with motion and perception modules to convert high-level decisions into robot behaviour, for example by initiating a navigation step, requesting a measurement, or activating a precision routine.
- **System state machine:** The top-level mission controller can be represented by a System FSM supervising global sequencing, controlling transitions between main mission phases, evaluating system conditions, and maintaining error and emergency procedures.
- **Sub-state machines:** Sub-FSMs can represent specialised controllers dedicated to structured procedures. This supports decomposition of complex behaviours into reusable components while remaining coordinated under a parent FSM.

Mission progression commonly follows a pattern in which external modules update mission conditions through status setters (e.g., `setStatus`), and a subsequent event such as `next` triggers transition evaluation. This approach provides a clear mapping between mission conditions and internal state evolution without requiring an excessive number of event types. The following simplified example illustrates the SML rule structure aligned with Figure 3.1:



```

1 namespace sml = boost::sml;
2
3 struct Next {};
4 struct Idle {};
5 struct Navigating {};
6 struct Procedure {};
7
8 auto has_good_localization = [](const auto& ctx) { return ctx.localization_ok;
9     ↪ };
10 auto start_navigation      = [](auto& ctx) { ctx.nav.start(); };
11 auto start_procedure       = [](auto& ctx) { ctx.proc.begin(); };
12
13 struct MissionSm {
14     auto operator()() const {
15         using namespace sml;
16         return make_transition_table(
17             * state<Idle> + event<Next> [ has_good_localization ] / start_navigation
18             ↪ = state<Navigating>,
19             state<Navigating> + event<Next> / start_procedure = state<Procedure>
20         );
21     }
22 };

```

Code 3.1: Illustrative Boost SML transition rules with event, guard, and action.

Together, these elements provide a deterministic, modular, and verifiable foundation for implementing mission-level logic. Chapter 5 applies these mechanisms to the SPS survey procedure, including constrained behaviours and recovery policies.

#### 3.4 Visual Servoing with Fiducial Markers

Visual servoing was not available as a dedicated module in the CRF before this work. As a consequence, operations requiring tight alignment, such as safety-gate traversal and docking at the charging station, were performed under manual control, despite the availability of motion controllers and camera acquisition pipelines.

To address this limitation, the SPS survey extensions introduce position-based visual servoing based on fiducial markers. *ArUco* markers provide stable geometric references at key locations such as safety gates and charging stations, enabling repeatable alignment under challenging lighting conditions. The resulting perception-to-control loop supports the precision required in constrained tunnel areas while remaining compatible with CRF motion-control abstractions. The detailed software realisation of this subsystem, including marker pose estimation, PBVS control, and filtering, is presented in Chapter 4.

#### 3.5 Design Rationale and Integration

The design choices described above are guided by the operational constraints of the SPS tunnel and CERN requirements for safety, modularity, and repeatability. The CRF provides a proven baseline for modular mission integration and deterministic FSM execution. However, the SPS survey procedure exposes limitations of purely teleoperated execution and nominal navigation assistance, particularly during constrained gate and docking phases and during long-duration operation with intermittent communication and localisation degradation.

The extensions introduced for the SPS radiation surveys therefore build upon CRF mechanisms

while adding mission-level sequencing, structured recovery, and precision alignment capabilities. Together, these components form a cohesive and extensible architecture that supports long-term supervised autonomous radiation surveys. Detailed discussions of the Mission Manager structure and the adopted precision manoeuvring approach are provided in Chapter 5 and Section 4.1.3.3, respectively.

## Chapter 4. Improved and New System Components

This chapter describes the software components that allow fully autonomous radiation surveys within the CRF, building on the basis described in Chapter 3. It focuses on the subsystems responsible for perception, localization, motion control, and visual servoing, highlighting their interaction within the overall mission architecture.

For accurate alignment with visual targets such fiducial markers, particular attention is given to the **TargetVisualServoing** module, which implements a reusable Position-Based Visual Servoing controller. Its design, perception processing, control formulation, and experimental validation under realistic tunnel conditions are described in the following sections.

### 4.1 Overview of the Visual Servoing Subsystem

Chapter 3 introduced the use of PBVS with fiducial markers from a system level perspective and motivated its role within the SPS radiation survey mission. This section refines that description by summarising the concrete software realisation of the visual servoing subsystem implemented in this thesis.

Within the CRF, the visual servoing functionality is decomposed into two main software modules:

- **MarkerPoseDetector:** Provides per frame six degree of freedom (6 DoF) pose estimates of the target marker from camera images, implementing a refactored *ArUco* based detection pipeline compatible with the new perception–control architecture.
- **TargetVisualServoing:** Implements the PBVS control loop, including pose filtering, motion prediction and the generation of Cartesian velocity commands that are sent to the motion controller through the **IMotionController** interface.

Both modules interact through a well defined interface and rely on existing CRF components such as the camera drivers and the motion controller. The following sections describe each element in detail: first the **MarkerPoseDetector**, which was adapted and modularised for this work, and then the **TargetVisualServoing** module, which was designed and implemented in the framework of this thesis.

#### 4.1.1 Marker Pose Detector

The **MarkerPoseDetector** class estimates the 6-DoF pose of fiducial markers from camera images. Although this functionality already existed within the CRF, it was refactored and modularized in this work to ensure full compatibility with the perception–control architecture used by the visual servoing subsystem. The redesigned implementation fits with the **ITargetPoseDetector** interface, which makes it possible for the system to easily switch between detectors.

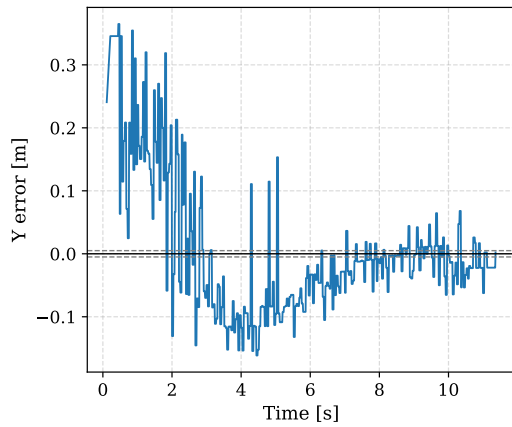
The detector uses known marker geometry and calibrated camera intrinsics to extract marker poses from individual frames using OpenCV’s *ArUco* library. Although this method gives precise static estimates, it may display discontinuities and high-frequency jitter when in motion, especially when the camera or robot is moving. Such instability can introduce noise into the PBVS control loop, as small variations in the measured pose directly impact the computed velocity commands.

In the SPS tunnel, these effects become more pronounced due to operational conditions that are not present in laboratory settings. During experiments with the MIRA platform, three recurring limitations were observed.

First, **pose jitter** appears even under slow motion, caused by small vibrations of the mobile base and by frame to frame variability in corner localization. Since the PBVS controller transforms instantaneous pose errors into Cartesian velocity commands, this jitter directly produces oscillatory control outputs and occasional sharp peaks.

Second, **lighting sensitivity** affects detection quality. Non uniform illumination, reflections on metallic surfaces, and local overexposure can lead to intermittent misdetections and abrupt pose jumps, despite the marker remaining static in the environment.

Third, **temporal occlusions** occur frequently in realistic manoeuvres, especially during safety gate crossing and docking, where the marker can be partially or fully hidden by structural elements or by the robot geometry itself. In these intervals, raw per frame detection provides no valid reference, which can destabilize the closed loop behaviour if the controller does not include a continuity mechanism.



(a) Pose jitter in the estimated marker pose under motion.



(b) Sensitivity to lighting changes and reflections in SPS.



(c) Temporary occlusions leading to intermittent detections.

Figure 4.1: Main constraints observed in *ArUco* based detections during MIRA operation in SPS.

Unlike other perception pipelines that include temporal filtering or motion prediction, the **MarkerPoseDetector** is explicitly designed to function on a per-frame basis. The

**TargetVisualServoing** module, which is explained in the next section, handles all filtering, smoothing, and temporal prediction while providing instantaneous 6-DoF pose measurements of identified markers. This separation of perception and control ensures modularity, decreases coupling between components, and allows the same detector to be reused in other contexts where real-time feedback or different filtering strategies may be required.

#### 4.1.2 Kalman Filter for Marker Pose Tracking

To achieve smooth and temporally consistent pose estimation during visual servoing, this work integrates a linear Kalman filter. By fusing incoming *ArUco* measurements with a constant velocity motion model, this estimator provides a continuous 6 DoF camera–marker transformation, guaranteeing strong feedback even in the case of transient marker loss or motion blur.

The implementation is based on the open source header only Kalman library by Markus Herb [30], affiliated with the Technical University of Munich and released under an MIT license. The library supports both linear and nonlinear filters, such as Extended and Unscented Kalman Filters, and offers a lightweight, flexible C++ framework for creating state estimators using template-based models.

Its modular design preserved real-time performance and compatibility with current perception and control interfaces, enabling smooth integration into the CERN Robotic Framework.

Two practical considerations that go beyond the fundamental linear filter are essential to this implementation and are discussed in detail. First, the transition matrix  $\mathbf{A}(\Delta t)$  is updated at runtime since the state transition employs the measured elapsed time between updates rather than a predetermined sampling interval. Second, a statistical gate based on the innovation’s *Mahalanobis* distance excludes erroneous *ArUco* measurements before they reach the correction stage, ensuring outlier resilient operation.

##### 4.1.2.1 Dynamic and Observation Models.

The dynamic system is modeled under a constant velocity assumption, which is suitable for the smooth and short term motion typically seen in visual servoing. The state stacks both the 6 DoF pose, position and orientation, and their corresponding velocities:

$$\mathbf{x}_t = \begin{bmatrix} \mathbf{p}_t \\ \boldsymbol{\theta}_t \\ \dot{\mathbf{p}}_t \\ \dot{\boldsymbol{\theta}}_t \end{bmatrix} \in \mathbb{R}^{12}, \quad (4.1)$$

where  $\mathbf{p}_t = [x, y, z]^\top$  denotes position and  $\boldsymbol{\theta}_t = [\alpha, \beta, \gamma]^\top$  denotes orientation angles in radians.

The temporal evolution follows a linear constant velocity model with a measured time step. Rather than assuming a fixed sampling period, the transition matrix is refreshed at runtime with the actual elapsed time  $\Delta t = t_{\text{now}} - t_{\text{prev}}$  obtained from a monotonic clock:

$$\mathbf{x}_t = \mathbf{A}(\Delta t) \mathbf{x}_{t-1} + \mathbf{w}_t, \quad \mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \quad (4.2)$$

with

$$\mathbf{A}(\Delta t) = \begin{bmatrix} \mathbf{I}_6 & \Delta t \mathbf{I}_6 \\ \mathbf{0}_6 & \mathbf{I}_6 \end{bmatrix}. \quad (4.3)$$

This construction explicitly express constant velocity kinematics and automatically accounts jitter in the camera or processing pipelines, since  $\Delta t$  is measured at each cycle.

The *ArUco* detector, which provides the relative camera to marker transformation at each frame, estimates the 6D pose in direct correlation with the observation model:

$$\mathbf{z}_t = \mathbf{C} \mathbf{x}_t + \mathbf{v}_t, \quad \mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \quad (4.4)$$

with

$$\mathbf{C} = [\mathbf{I}_6 \quad \mathbf{0}_{6 \times 6}]. \quad (4.5)$$

This formulation follows the approach proposed by Kam *et al.* [31]. Here the orientation is expressed with Euler angles rather than quaternions, which simplifies computation and preserves local linearity for small angular motions, a valid approximation in short horizon visual servoing.

#### 4.1.2.2 Outlier Rejection via Mahalanobis Distance.

To avoid using incorrect *ArUco* detections that could distort the filter estimate, a simple statistical test is applied to the innovation before the correction step. The innovation and its covariance are defined as:

$$\mathbf{r}_t = \mathbf{z}_t - \mathbf{C} \hat{\mathbf{x}}_{t|t-1}, \quad \mathbf{S}_t = \mathbf{C} \mathbf{P}_{t|t-1} \mathbf{C}^\top + \mathbf{R}. \quad (4.6)$$

The Mahalanobis distance is then computed as:

$$d_M(\mathbf{r}_t) = \sqrt{\mathbf{r}_t^\top \mathbf{S}_t^{-1} \mathbf{r}_t}. \quad (4.7)$$

A measurement is accepted only if  $d_M(\mathbf{r}_t) < \tau_M$ ; otherwise, the update is skipped and the filter keeps the predicted state. In this work, the threshold is set to  $\tau_M = 3.0$ , which provides a conservative limit. Alternatively,  $d_M^2$  can be compared with the chi-square value  $\chi_{k,\alpha}^2$  for  $k = 6$  measured dimensions; for example,  $\chi_{6,0.95}^2 \approx 12.6$  corresponds to a distance of about 3.55. Using  $\tau_M = 3.0$  increases robustness by rejecting more outliers, even if a few valid measurements are also discarded.

#### 4.1.2.3 Covariance Estimation and Parameter Selection.

The covariance matrices used in this work were obtained primarily from experimental data, complemented with guidance from the literature on Kalman filtering for visual tracking and marker-based pose estimation [32, 33]. Unlike purely heuristic tuning, this procedure aims to ground the filter parameters on the actual noise characteristics observed in the system.

The measurement noise covariance matrix  $\mathbf{R}_{\text{base}}$  was estimated experimentally by recording the pose returned by the *ArUco* detector while the robot and the camera remained completely stationary for several seconds. From this dataset, the empirical covariance of the six measured pose components  $(e_x, e_y, e_z, \omega_x, \omega_y, \omega_z)$  was computed. The resulting diagonal terms were on the order of  $10^{-6}$ – $10^{-4}$  (in SI units), consistent with typical frame-to-frame jitter reported for monocular marker-based localization [31]. This confirmed that, under the illumination and sensing conditions of the experiments, the measurement noise is dominated by the vision pipeline rather than by robot motion.

The process noise covariance matrix  $\mathbf{Q}$  represents uncertainty in the assumed motion model. In this work, the state vector was defined as

$$\mathbf{x} = [\mathbf{p} \quad \boldsymbol{\theta} \quad \mathbf{v} \quad \boldsymbol{\omega}]^\top \in \mathbb{R}^{12}, \quad (4.8)$$

where  $\mathbf{p} \in \mathbb{R}^3$  and  $\boldsymbol{\theta} \in \mathbb{R}^3$  are the translational and rotational components of the relative pose (marker with respect to the camera), and  $\mathbf{v} \in \mathbb{R}^3$  and  $\boldsymbol{\omega} \in \mathbb{R}^3$  are their corresponding linear and angular velocities. A discrete constant-velocity model with variable sampling time  $dt$  was assumed, consistent with the acquisition rate of the camera frames.

To estimate the base process noise covariance  $\mathbf{Q}_{\text{base}}$  from real experiments, several closed-loop visual servoing runs were recorded while the robot moved from an initial misalignment to a final alignment with the target. For each run, the measured pose error signal  $(\mathbf{p}_k, \boldsymbol{\theta}_k)$  was sampled at the camera rate and used to compute discrete velocities and accelerations. Velocities were obtained by first-order finite differences,

$$\mathbf{v}_k \approx \frac{\mathbf{p}_k - \mathbf{p}_{k-1}}{dt}, \quad \boldsymbol{\omega}_k \approx \frac{\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}}{dt}, \quad (4.9)$$

and accelerations were computed by differentiating the velocities. To mitigate the amplification of high-frequency measurement noise introduced by numerical differentiation, the resulting acceleration signals were smoothed and outliers were rejected. In addition, accelerations were saturated using the robot task-space acceleration limits in order to discard non-physical spikes attributable to vision quantization and detection jitter.

The process noise was modelled as white Gaussian acceleration noise driving the constant-velocity system. Under this assumption, the process covariance has the standard form for each position-velocity pair,

$$\mathbf{Q}_i = \sigma_{a,i}^2 \begin{bmatrix} \frac{dt^3}{3} & \frac{dt^2}{2} \\ \frac{dt^2}{2} & dt \end{bmatrix}, \quad (4.10)$$

and the full  $\mathbf{Q} \in \mathbb{R}^{12 \times 12}$  is constructed by placing these blocks for the three translational axes and the three rotational axes, including the cross-terms between pose and velocity states. Using the experimentally estimated acceleration variances and the average sampling period ( $dt \approx 10$  ms), the resulting base process noise covariance matrix was

$$\mathbf{Q}_{\text{base}} = \begin{bmatrix} 3.74 \times 10^{-7} \mathbf{I}_3 & \mathbf{0} & 5.50 \times 10^{-5} \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & 1.74 \times 10^{-6} \mathbf{I}_3 & \mathbf{0} & 2.56 \times 10^{-4} \mathbf{I}_3 \\ 5.50 \times 10^{-5} \mathbf{I}_3 & \mathbf{0} & 1.08 \times 10^{-2} \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & 2.56 \times 10^{-4} \mathbf{I}_3 & \mathbf{0} & 5.03 \times 10^{-2} \mathbf{I}_3 \end{bmatrix}. \quad (4.11)$$

This matrix reflects the experimentally observed fact that the uncertainty associated with velocity states is substantially larger than that of pose states, which is consistent with a constant-velocity approximation of a real closed-loop motion. In practice, however, the absolute scale of  $\mathbf{Q}_{\text{base}}$  and  $\mathbf{R}_{\text{base}}$  may not perfectly capture all unmodelled effects present during closed-loop operation (e.g., residual detection outliers, discretization effects, and model mismatch). For this reason, the final covariances were obtained by applying scalar tuning factors that preserve the experimentally estimated structure while adjusting the relative confidence between prediction and measurement:

$$\mathbf{Q} = \alpha_Q \mathbf{Q}_{\text{base}}, \quad \mathbf{R} = \alpha_R \mathbf{R}_{\text{base}}. \quad (4.12)$$

Increasing  $\alpha_Q$  increases the predicted uncertainty between frames and therefore increases the Kalman gain, making the filter rely more strongly on incoming measurements. Conversely,

increasing  $\alpha_R$  increases the assumed measurement uncertainty and shifts the filter behaviour towards model prediction. Since the constant-velocity model is only a coarse approximation of the relative motion during closed-loop visual servoing, the tuning objective was to prioritise real detections whenever they are consistent, while attenuating high-frequency jitter without introducing excessive lag.

To assess the impact of these scaling choices, several Kalman filter configurations were evaluated under strictly identical experimental conditions. In all trials, the robot started from the same initial pose relative to the marker and executed the same alignment manoeuvre using identical PBVS controller gains. As a result, differences in the recorded error profiles can be attributed to the filtering configuration rather than to variations in the control law or initial conditions.

Figure 4.2 compares the evolution of the position and orientation error components for a reduced set of representative configurations, including the unfiltered case as a baseline. The unfiltered measurements exhibit noticeable high-frequency fluctuations, particularly in the orientation components, which are characteristic of raw *ArUco*-based pose estimation. Although the overall convergence trend is preserved, these oscillations degrade the smoothness and readability of the error signals.

Configurations that favour the prediction model yield smoother trajectories and attenuate measurement noise more aggressively. The trade-off is reduced responsiveness: during transient phases, corrections can be delayed when the error evolves rapidly, as the estimate follows the model more closely than the incoming detections.

Based on the qualitative comparison shown in Fig. 4.2, the configuration ( $\alpha_Q = 100$ ,  $\alpha_R = 0.01$ ) was selected. This scaling lies in an intermediate regime where neither prediction nor measurements dominate the estimation process. Visually, it preserves the reactivity of the unfiltered signal while substantially reducing high-frequency jitter, without introducing noticeable lag or over-smoothing artefacts.

For these reasons, the configuration ( $\alpha_Q = 100$ ,  $\alpha_R = 0.01$ ) was retained for all subsequent experiments involving Kalman-filtered and predictive feedback.

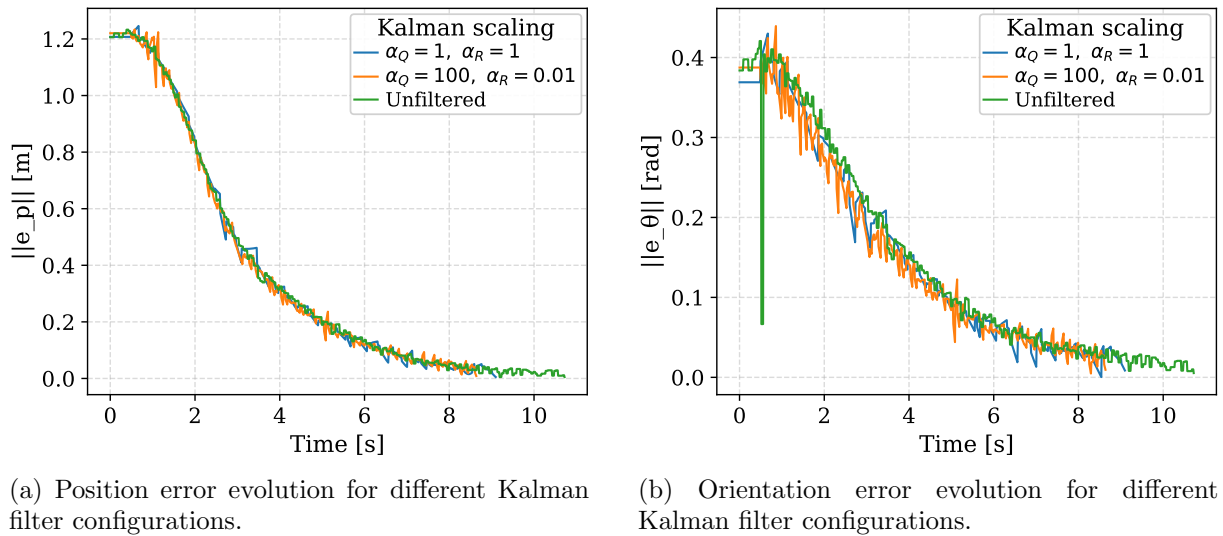


Figure 4.2: Comparison of Kalman filter configurations for marker pose estimation.



**4.1.2.4 Alternative Filtering Strategies.**

To check how temporal filtering affects PBVS stability, three different approaches were implemented. They differ in how the *ArUco* pose measurements are processed before being used by the PBVS controller:

**(a) Raw visual estimation:** This baseline approach sends each pose estimated by the `MarkerPoseDetector` directly to the PBVS controller at the camera frame rate (about 30 Hz). It prioritizes responsiveness and minimal delay, but measurement jitter and occasional dropouts are not mitigated. When the marker is lost, the controller stops receiving valid updates until the detection resumes.

---

**Algorithm 1** Raw visual estimation loop

---

```

1: while servoing active do
2:   frame  $\leftarrow$  camera.read()
3:   pose  $\leftarrow$  detectArucoPose(frame)
4:   if pose valid then
5:     PBVSController.update(pose)
6:   end if
7: end while

```

---

**(b) Continuous Kalman filtering:** In this configuration, the PBVS controller receives only the filtered pose produced by the Kalman estimator. The filter runs continuously, predicting the state at every iteration and correcting it when a new measurement is available. Even during short detection losses, the controller keeps receiving predicted poses, ensuring smooth and consistent feedback.

---

**Algorithm 2** Kalman filtering with continuous prediction

---

```

1: while servoing active do
2:   predictState()
3:   if marker detected then
4:     measurement  $\leftarrow$  detectArucoPose(frame)
5:     correctState(measurement)
6:   end if
7:   pose  $\leftarrow$  getFilteredPose()
8:   PBVSController.update(pose)
9: end while

```

---

**(c) Hybrid predictive filtering:** This variant combines the reactivity of raw feedback with the continuity of Kalman prediction. When the marker is confidently detected, the raw pose is sent directly to the controller to minimize delay. If the marker is lost or rejected by the Mahalanobis gate, the system switches to the Kalman prediction, which propagates the last valid state. This prevents sudden discontinuities in control and maintains stable behaviour during brief occlusions.

---

**Algorithm 3** Hybrid predictive filtering loop

---

```

1: while servoing active do
2:   if marker detected and valid then
3:     pose  $\leftarrow$  detectArucoPose(frame)
4:     correctState(pose)
5:     PBVSController.update(pose)
6:   else
7:     predictState()
8:     pose  $\leftarrow$  getFilteredPose()
9:     PBVSController.update(pose)
10:  end if
11: end while

```

---

In summary, the raw mode provides the most immediate response but is sensitive to visual noise, the Kalman mode ensures smoothness and continuity by relying solely on filtered estimates, and the hybrid mode achieves the best compromise by using prediction only when visual information is unavailable or unreliable.

#### 4.1.2.5 Robustness Evaluation and Occlusion Test.

In order to analyze how the visual feedback behaved when the marker was momentarily removed, a number of controlled occlusion experiments were carried out. The PBVS controller was forced to rely on either direct *ArUco* measurements (raw mode) or the prediction step of a linear Kalman filter (classical mode) because the fiducial marker was purposefully covered for a brief period of time in these tests (shown by the red shaded region in Figure 4.3).

This causes the visual error  $\mathbf{e}$  to suddenly discontinue, which would cause the PBVS controller to deliver huge or unpredictable velocity commands. This behavior is dangerous on a real robot since the controller unexpectedly loses the reference, particularly if the robot is crossing a gate or docking into a charging station.

The last valid measurement is propagated using a constant velocity motion model with the classical Kalman configuration. The orange trace indicates that the filter output remains constant close to the pre-occlusion value throughout the occlusion interval. This keeps the PBVS loop from becoming unstable right away and continues to provide the controller a realistic position until the visual feedback comes back. This behavior is definitely better than the raw technique for short occlusions.

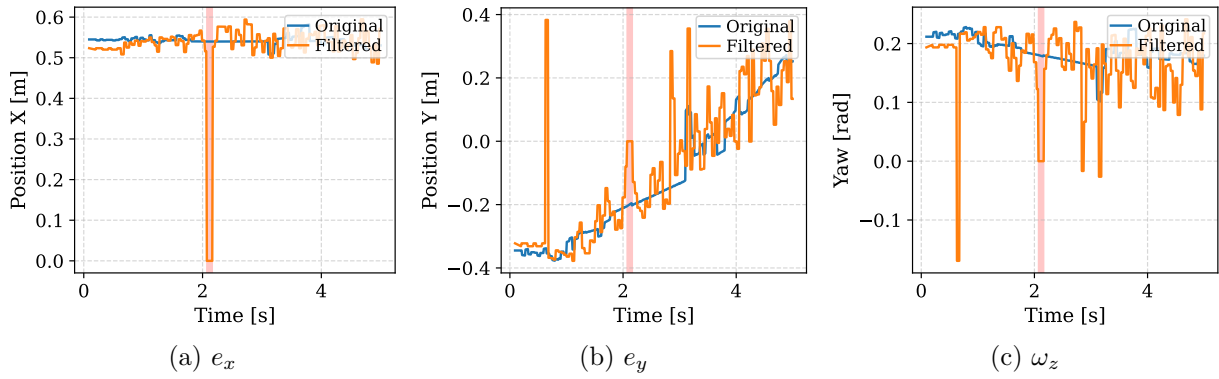


Figure 4.3: PBVS errors during a controlled marker occlusion. Orange: original. Blue: filtered (zeroed when lost). Red bands: marker lost.

These experiments showed that, while Kalman filtering already improves robustness to short occlusions, pure raw visual feedback is too fragile because losing the marker means losing the reference altogether. This observation motivated the introduction of the predictive extension in the next subsection, whose goal is to keep the pose temporally aligned with the high rate control loop even when the visual updates are intermittent.

### 4.1.3 Target Visual Servoing Module

This project also included the whole development of the visual servoing subsystem, including its filtering technique, control implementation, and perception–control architecture. This covers the entire PBVS control loop, and the design and implementation of the `TargetVisualServoing` class. Every component was created from the ground up and added as a reusable, platform-independent module to the CERN Robotic Framework. The resulting subsystem supports both eye-in-hand and eye-to-hand configurations across several robotic platforms and offers thread-safe, real-time visual feedback control.

The `IVisualServoing` interface defines the essential API of any visual servoing module, including methods to `start()` a servoing session toward a given marker, `stop()` execution, query the current `getState()`, and verify convergence with `isTargetReached()`. It is possible to deploy many visual servoing systems reliably inside the same software ecosystem by following this common interface.

#### 4.1.3.1 Architecture and Core Components

The perception and control procedures necessary for PBVS operation are coordinated by the `TargetVisualServoing` module. It keeps track of its main dependencies: the camera driver (`ICamera`), the robot’s motion interface (`IMotionController`), and a `MarkerPoseDetector`. It runs two concurrent threads internally: one for the control loop and the other for camera acquisition and marker pose estimation. To maintain thread safety and synchronization, these threads exchange information via shared variables that are guarded by mutexes.

The `ITargetPoseDetector` interface abstracts marker detection. Different perception backends (such *AprilTags* or trained models) can be transparently changed without changing the control logic thanks to this modular design.

The module includes an internal filtering layer for pose estimation, which is explained in detail in Section 4.1.2, to ensure consistent and continuous feedback. Even in the event of temporary visual data loss or partial occlusions, this layer guarantees constant tracking.

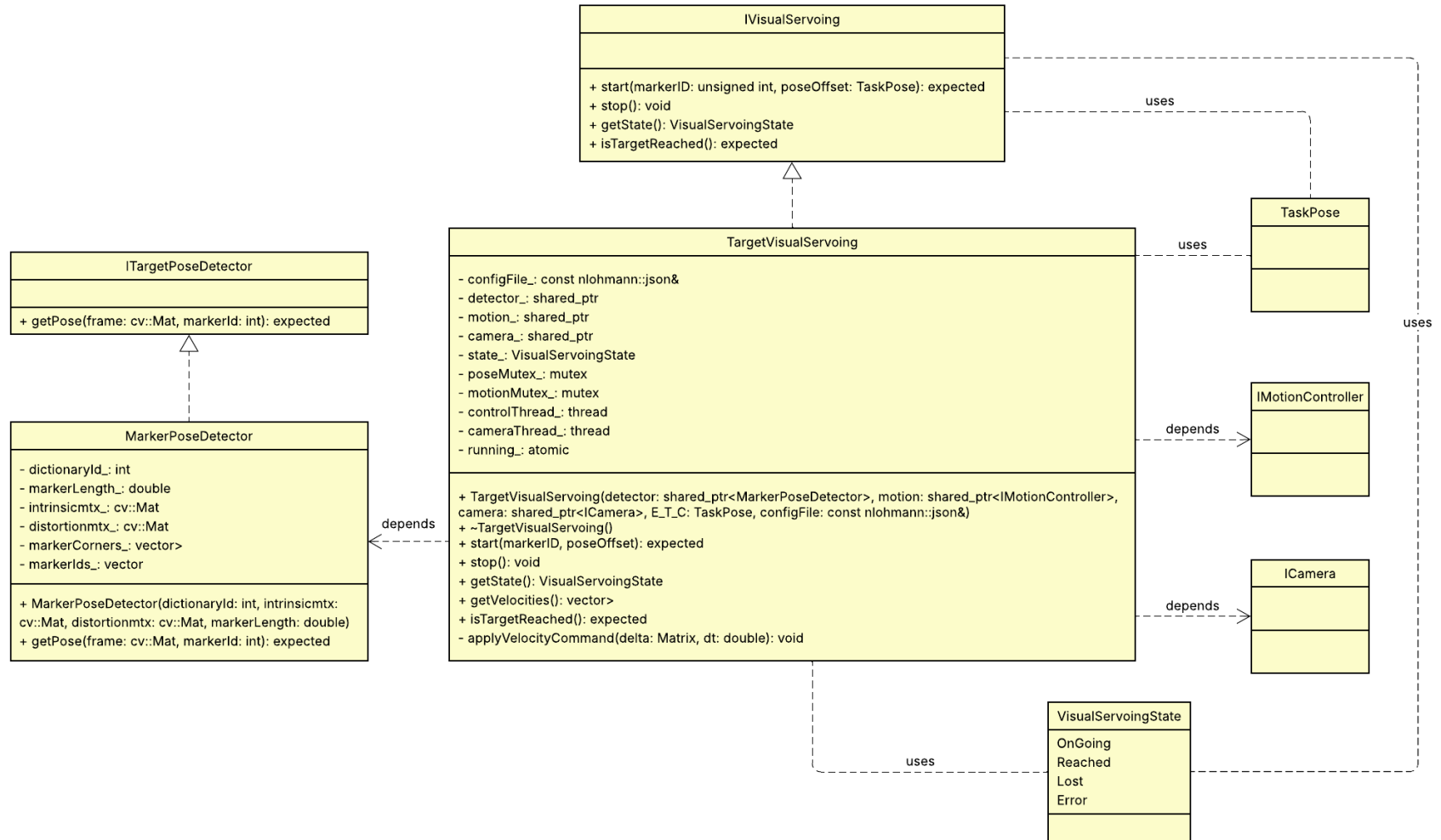


Figure 4.4: UML diagram of the **TargetVisualServoing** module, showing its integration with detector, camera, and motion controller interfaces.

#### 4.1.3.2 Configuration File and Parameterization

The `TargetVisualServoing` module is configured through a dedicated JSON block that enables reproducible deployment across robots and sensing setups without modifying the control implementation. The configuration externalizes all parameters required by the perception pipeline, the geometric frame chain, and the PBVS controller, so that calibration, constraints, and tuning can be adjusted without touching the core code.

1. **Camera parameters:** This section contains the perception prerequisites required by the pose detector. It typically includes the intrinsic calibration (camera matrix) and distortion coefficients, the *ArUco* dictionary identifier, and the stream endpoints consumed by the `ICamera` driver. These parameters define how pose measurements are produced and directly influence the stability and accuracy of the feedback signal.
2. **Common servoing parameters:** This subsection defines setup-dependent parameters that are usually shared across multiple servoing tasks:
  - **Controlled axes:** boolean flags enabling or disabling individual components of the commanded twist  $\boldsymbol{\nu} = [v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]^\top$ . This allows constraining motion (for instance planar control or yaw-only alignment) and improves safety when some DoFs must remain fixed.
  - **Camera extrinsics:** the end-effector to camera transform  ${}^E\mathbf{T}_C$  (Equation 4.18), required to express the visually estimated target in the end-effector and base frames.
  - **Cartesian velocity limits:** per-axis saturation bounds used to clip  $\boldsymbol{\nu}$  into a safe command  $\boldsymbol{\nu}_{\text{cmd}}$ , preventing aggressive motions due to transient estimation errors or large initial offsets.
3. **Task-specific parameters:** Each servoing mode can define its own control tuning and termination criteria:
  - **PID gains** ( $K_p, K_i, K_d$ ) applied to the 6D pose error (Equation 4.21).
  - **Convergence tolerances** ( $\varepsilon_t, \varepsilon_r$ ) used by `isTargetReached()` to stop servoing when both translation and rotation errors fall below thresholds.
  - **Optional overrides** of velocity limits or transforms whenever a specific task requires stricter constraints.
4. **Marker definitions and target poses:** Visual tasks also specify the marker used as reference by the `ITargetPoseDetector`, including its identifier and physical size. Additionally, a task can define one or more target poses relative to the marker. These poses correspond to the fixed offset  ${}^M\mathbf{T}_{Ta}$  used in Equation 4.16, allowing the robot to converge to a desired alignment pose rather than the marker origin.

Listing 4.1 shows a simplified configuration example for a task that controls  $\{v_x, v_y, \omega_z\}$  only, uses a calibrated  ${}^E\mathbf{T}_C$  transform, and saturates the commanded Cartesian twist with task-specific limits.

```

1 {
2   "PID": {
3     "Kp": 0.55,
4     "Ki": 0.0,
5     "Kd": 0.005
6   },
7
8   "Tolerance": {
9     "Position": 0.005,
10    "Orientation": 0.01
11  },
12
13  "ControlledAxes": {
14    "Vx": true,
15    "Vy": true,
16    "Vz": false,
17    "Wx": false,
18    "Wy": false,
19    "Wz": true
20  },
21
22  "E_T_C": {
23    "Translation": [0.0, 0.0, 0.0],
24    "Rotation": [-1.57, 0.0, -1.57]
25  },
26
27  "TaskVelocityLimits": [0.30, 0.30, 0.10, 0.10, 0.10, 0.50]
28 }

```

Code 4.1: Example JSON configuration for a **TargetVisualServoing** task.

Table 4.1 summarizes the most relevant fields and their role in the PBVS loop.

| Field                 | Meaning and usage                                                                                                                                      |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| PID.Kp, Ki, Kd        | Cartesian PID gains applied to the 6D pose error $\mathbf{e}$ to generate $\boldsymbol{\nu}$ (Equation 4.21).                                          |
| Tolerance.Position    | Translational convergence threshold $\varepsilon_t$ (meters) used in $\ \mathbf{e}_t\  < \varepsilon_t$ .                                              |
| Tolerance.Orientation | Rotational convergence threshold $\varepsilon_r$ (radians) used in $\ \mathbf{e}_r\  < \varepsilon_r$ .                                                |
| ControlledAxes.*      | Per-axis enable flags for $\{v_x, v_y, v_z, \omega_x, \omega_y, \omega_z\}$ , used to enforce motion constraints.                                      |
| E_T_C                 | End-effector to camera transform ${}^E\mathbf{T}_C$ used to propagate target poses across frames (Equation 4.18).                                      |
| TaskVelocityLimits    | Saturation bounds for $\boldsymbol{\nu}_{\text{cmd}}$ in the order $\{v_x, v_y, v_z, \omega_x, \omega_y, \omega_z\}$ , ensuring safe bounded commands. |

Table 4.1: Main configuration parameters of the **TargetVisualServoing** module and their mapping to the PBVS loop.

#### 4.1.3.3 Pose-Based Visual Servoing Control

The second component of the `TargetVisualServoing` module implements a Cartesian PBVS scheme. At each iteration, the most recent marker pose is transformed through the chain of reference frames (marker  $\rightarrow$  camera  $\rightarrow$  end-effector  $\rightarrow$  base), allowing the control law to operate entirely in Cartesian space. The 6D error between the desired and current end-effector poses is computed and mapped into a twist vector containing both translational and rotational components.

#### Frame Composition

Each pose in the system is represented as a `TaskPose` object encapsulating translation  $\mathbf{t} \in \mathbb{R}^3$  and rotation, stored as either a quaternion  $\mathbf{q}$  or rotation matrix  $\mathbf{R}$ :

$${}^A\mathbf{P}_B = \begin{bmatrix} {}^A\mathbf{R}_B & {}^A\mathbf{t}_B \\ \mathbf{0} & 1 \end{bmatrix}. \quad (4.13)$$

At runtime, the camera thread estimates the pose of the marker in the camera frame:

$${}^C\mathbf{T}_M = \begin{bmatrix} {}^C\mathbf{R}_M & {}^C\mathbf{t}_M \\ \mathbf{0} & 1 \end{bmatrix}, \quad (4.14)$$

and a fixed offset defines the relative pose between the marker and the task target:

$${}^M\mathbf{T}_T = \begin{bmatrix} {}^M\mathbf{R}_{Ta} & {}^M\mathbf{t}_{Ta} \\ \mathbf{0} & 1 \end{bmatrix}. \quad (4.15)$$

The target pose in the camera frame is thus:

$${}^C\mathbf{T}_{Ta} = {}^C\mathbf{T}_M {}^M\mathbf{T}_{Ta}. \quad (4.16)$$

Using the known camera–end-effector calibration,

$${}^E\mathbf{T}_C = \begin{bmatrix} {}^E\mathbf{R}_C & {}^E\mathbf{t}_C \\ \mathbf{0} & 1 \end{bmatrix}, \quad (4.17)$$

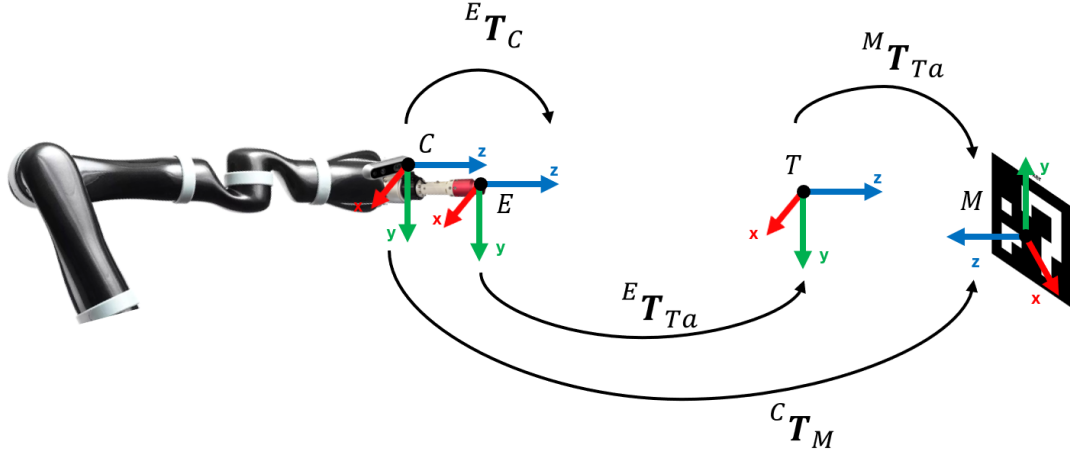
the target pose can be expressed in the end-effector frame:

$${}^E\mathbf{T}_{Ta} = {}^E\mathbf{T}_C {}^C\mathbf{T}_{Ta}. \quad (4.18)$$

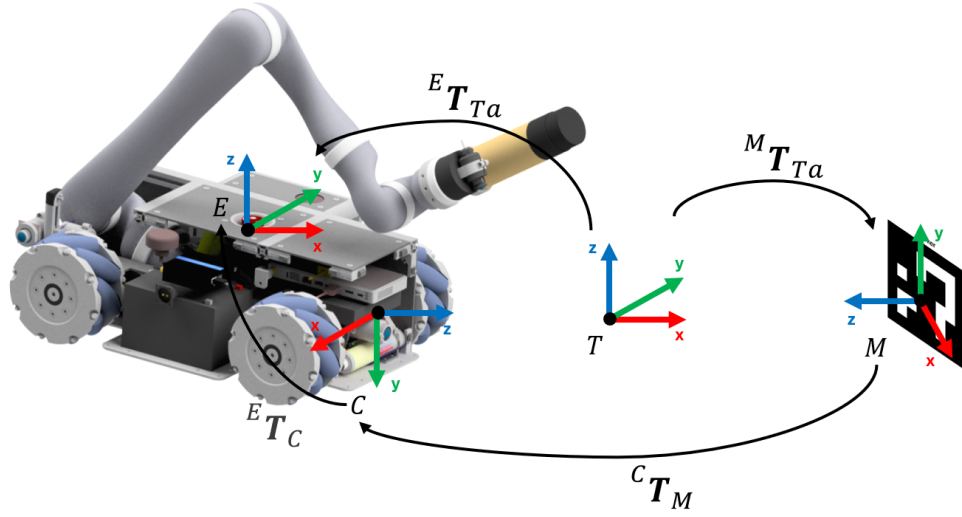
Finally, the current end-effector pose obtained from the motion controller allows expressing the target in the robot base frame:

$${}^B\mathbf{T}_{Ta} = {}^B\mathbf{T}_E {}^E\mathbf{T}_{Ta}. \quad (4.19)$$





(a) Kinova Gen2



(b) MIRA platform

Figure 4.5: Reference frames involved in the PBVS control loop for both setups. The transformations  ${}^M\mathbf{T}_{Ta}$  and  ${}^E\mathbf{T}_C$  are updated or calibrated depending on the robot configuration.

### Control Law

The control objective is to minimize the 6D pose error between the current and desired end-effector poses, both expressed in the base frame:

$$\mathbf{e} = \begin{bmatrix} \mathbf{e}_t \\ \mathbf{e}_r \end{bmatrix} = \begin{bmatrix} {}^B\mathbf{t}_T - {}^B\mathbf{t}_E \\ \frac{1}{2} ({}^B\mathbf{R}_E^\top {}^B\mathbf{R}_{Ta} - {}^B\mathbf{R}_{Ta}^\top {}^B\mathbf{R}_E)^\vee \end{bmatrix} \quad (4.20)$$

In practice, this computation is performed using the CRF quaternion-based distance function:

$$\mathbf{e} = \text{byQuaternion}(\text{base}, \text{E\_P\_T}, \text{true}),$$

which provides a smooth and singularity-free orientation error representation.

A PID controller regulates the Cartesian error according to:

$$\boldsymbol{\nu} = K_p \mathbf{e} + K_i \int \mathbf{e}(t) dt + K_d \frac{d\mathbf{e}}{dt}, \quad (4.21)$$

where  $\boldsymbol{\nu} = [v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]^\top$  represents the commanded end-effector velocity. The resulting command is saturated for safety:

$$\boldsymbol{\nu}_{\text{cmd}} = \text{clip}(\boldsymbol{\nu}, \boldsymbol{\nu}_{\text{min}}, \boldsymbol{\nu}_{\text{max}})$$

The command  $\boldsymbol{\nu}_{\text{cmd}}$  is sent to the **MotionController**, which resolves the robot Jacobian and converts Cartesian velocities into joint velocities:

$$\dot{\mathbf{q}} = \mathbf{J}^\dagger \boldsymbol{\nu}_{\text{cmd}}, \quad (4.22)$$

where  $\mathbf{J}$  is the robot Jacobian and  $\mathbf{J}^\dagger$  its damped pseudoinverse. This abstraction layer isolates the visual servoing logic from robot-specific kinematics, allowing the same module to operate seamlessly across multiple robotic platforms.

The control loop runs continuously, checking the freshness of the last valid pose and stopping only when both translational and rotational errors fall below the predefined convergence thresholds:

$$\|\mathbf{e}_t\| < \varepsilon_t, \quad \|\mathbf{e}_r\| < \varepsilon_r \quad (4.23)$$

Figure 4.6 illustrates the overall visual servoing architecture, where the estimated pose closes the control loop. The visual estimator fuses *ArUco* detections with a constant-velocity motion model, rejecting outliers through a Mahalanobis gate. In the hybrid predictive configuration, the state is further propagated to the control timestamp to maintain temporal alignment between sensing and actuation.

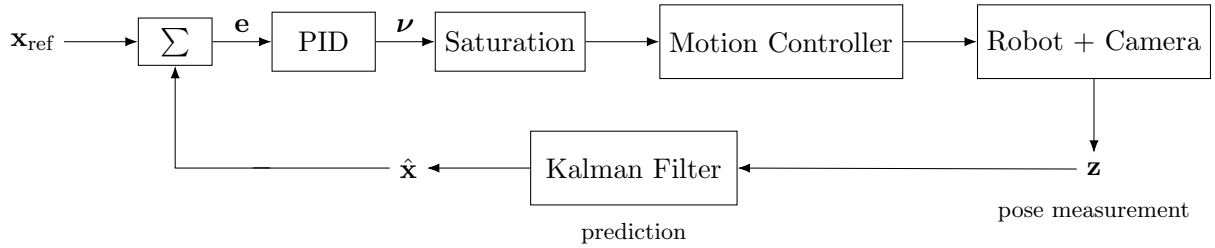


Figure 4.6: Closed loop of the PBVS controller with internal Kalman estimation and Mahalanobis outlier rejection. In the hybrid predictive mode, the estimated state is propagated to the control timestamp, ensuring temporally consistent feedback.

**Execution Flow**

The overall behaviour of the module is summarised in Algorithm 4, which highlights the interaction between the perception and control threads.

---

**Algorithm 4** Simplified execution flow of the `TargetVisualServoing` module

---

```

1: Initialize: reset PID terms, timers, Kalman filter, and logs
2: Start camera thread:
3: while active do
4:   frame  $\leftarrow$  camera.read()
5:   pose  $\leftarrow$  detectArucoPose(frame)
6:   if pose valid then
7:     updatePose(pose)
8:   else
9:     predictPose()
10:  end if
11: end while
12: Start control thread:
13: while active do
14:    $\mathbf{e} \leftarrow \text{computePoseError}(\text{target}, \text{current})$ 
15:    $\boldsymbol{\nu} \leftarrow K_p \mathbf{e} + K_i \int \mathbf{e} + K_d \dot{\mathbf{e}}$ 
16:    $\boldsymbol{\nu}_{\text{cmd}} \leftarrow \text{clip}(\boldsymbol{\nu}, \boldsymbol{\nu}_{\text{min}}, \boldsymbol{\nu}_{\text{max}})$ 
17:   applyVelocityCommand( $\boldsymbol{\nu}_{\text{cmd}}$ )
18:   if  $\|\mathbf{e}_t\| < \varepsilon_t$  and  $\|\mathbf{e}_r\| < \varepsilon_r$  then
19:     stop servoing
20:   end if
21: end while

```

---

This structure enables continuous closed-loop control driven by real-time visual feedback, ensuring smooth convergence and stable behaviour even under moderate visual interruptions.

**4.2 Experimental Validation on the Kinova Gen2 Arm**

A preliminary set of PBVS experiments was carried out on a Kinova Gen2 robotic arm as an additional validation to the main experimental campaign on MIRA (Section 4.3) in order to confirm the accuracy of the `TargetVisualServoing` implementation and to obtain an initial gain tuning reference under controlled laboratory conditions. Before the controller was deployed on the target system, its closed-loop behavior was studied using the Kinova platform, which operated in an eye-in-hand configuration.

A key difference with respect to the MIRA evaluation is that these Kinova tests relied directly on raw *ArUco* pose measurements as visual feedback, without Kalman filtering or prediction. This makes it possible to better understand the impact of proportional and derivative action under unfiltered visual feedback and isolates the controller reaction to immediate measurement noise.



Figure 4.7: Experimental setup used for PBVS validation on the Kinova Gen2 arm. A fiducial marker is mounted on a fixed support, and an Intel RealSense camera is attached to the robot end-effector in an eye-in-hand configuration.

The validation was executed in two stages. First, the PBVS loop was tested from multiple initial arm configurations relative to the same marker to confirm consistent convergence and correct qualitative behaviour across the workspace. Second, a controlled tuning sweep was performed by repeating the experiments from a single fixed initial pose while adjusting the controller gains. Across this second stage, camera calibration, lighting conditions, marker pose, and control rate were kept constant, and only the controller parameters were modified between runs.

For each run, the full 6D pose error was logged, including Cartesian position components  $(x, y, z)$  and orientation components (roll, pitch, yaw).

The tuning sweep considered a nominal proportional gain  $K_p = 0.05$  and evaluated the effect of derivative action on top of that baseline. The results are summarised in Table 4.2. In addition, for the selected configuration, the individual position and orientation components are shown in Figure 4.8.

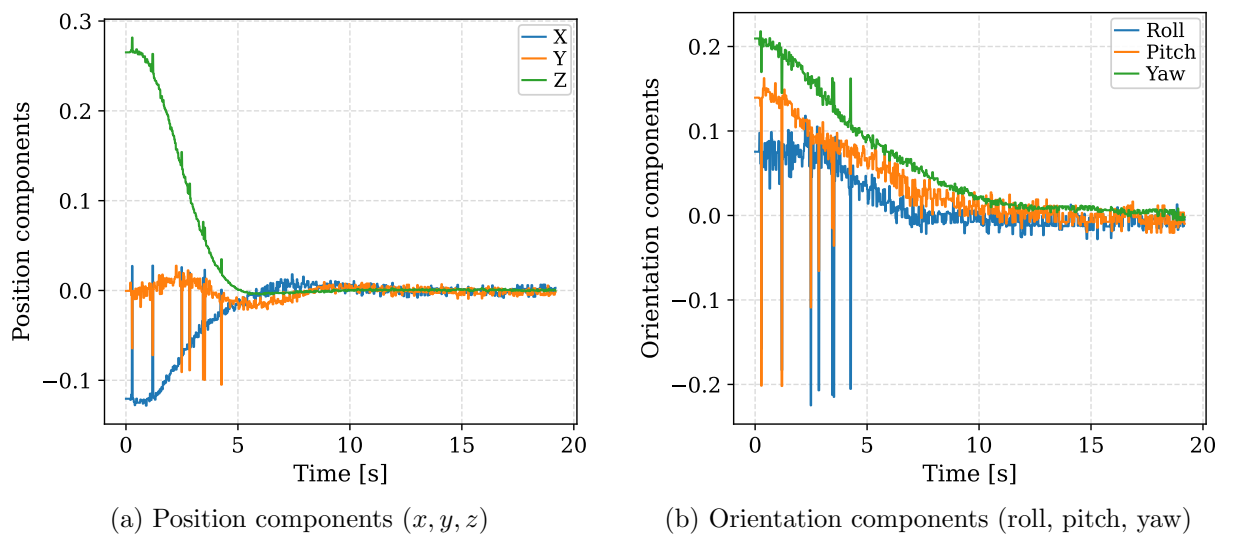


Figure 4.8: Kinova Gen2 PBVS validation using raw *ArUco* pose feedback (no Kalman filtering). Illustrative traces shown for the selected PD configuration ( $K_p = 0.05$ ,  $K_d = 0.02$ ).

Table 4.2: Performance under raw *ArUco* feedback on the Kinova Gen2 arm (no Kalman filtering).  $T_s$  is defined as the first time the 6D error enters the acceptance range.

| Controller           | $K_p$       | $K_d$       | $T_s$ [s]    | ISE          | Remarks                                   |
|----------------------|-------------|-------------|--------------|--------------|-------------------------------------------|
| P (baseline)         | 0.05        | 0.00        | 16.46        | 0.437        | Reference proportional behaviour          |
| <b>PD (selected)</b> | <b>0.05</b> | <b>0.02</b> | <b>19.18</b> | <b>0.429</b> | <b>Lowest ISE among tested candidates</b> |
| PD (higher $K_d$ )   | 0.05        | 0.03        | 18.83        | 0.467        | Higher accumulated error                  |

Table 4.2 provides a quantitative comparison of representative gain configurations under raw *ArUco* feedback using ISE as the primary performance indicator and  $T_s$  as the convergence speed measure. Under unfiltered visual measurements, ISE is particularly informative because it penalises large transient deviations strongly and captures the effect of residual jitter on the accumulated error without relying on overshoot-style quantities, which can be unreliable when the measurement signal is noisy.

Among the evaluated candidates, the PD controller with ( $K_p = 0.05$ ,  $K_d = 0.02$ ) achieves the lowest ISE (0.429), indicating the best overall reduction of squared pose error over the run. While its time to enter tolerance is longer than the purely proportional baseline,  $T_s$  remains within the same order of magnitude and the reduction in ISE reflects a more favourable global closed-loop behaviour under raw measurements. This is consistent with the representative traces in Figure 4.8, where the controller yields coherent, well-behaved position and orientation convergence profiles despite instantaneous noise in the *ArUco* pose estimation.

Therefore, the PD configuration ( $K_p = 0.05$ ,  $K_d = 0.02$ ) is retained as the preferred tuning reference from the Kinova validation. This initial baseline supports the subsequent evaluation on the MIRA platform described in Section 4.3, where filtering and prediction are introduced to improve robustness under harsher operating conditions.

### 4.3 Experimental Evaluation and Control Performance with MIRA

This section evaluates the PBVS loop under two visual feedback strategies: a hybrid predictive Kalman approach and a continuous Kalman-filtered approach, each combined with proportional (P) or proportional-derivative (PD) control. The analysis focuses on convergence speed and accumulated tracking quality, which are the most relevant indicators for reliable gate alignment and docking under noisy visual measurements.

For each run, the Cartesian position errors  $e_x$  and  $e_y$  and the yaw error  $e_\psi$  (corresponding to  $\omega_z$  in the controller) were logged. From these signals, the following metrics were computed:

- **Time to enter tolerance ( $T_s$ ):** the first instant at which the pose error enters the prescribed tolerance band. In this work,  $T_s$  directly measures convergence speed because the mission logic only requires the robot to reach the acceptance range.
- **Integrated squared error (ISE):** the time integral of the squared pose error norm,

$$\text{ISE} = \int \|e(t)\|^2 dt,$$

which captures overall tracking quality by penalising large errors more strongly than absolute-error metrics. This is particularly appropriate for PBVS with *ArUco* pose estimates, where brief error bursts and intermittent jitter can occur. Squaring the error

makes the metric sensitive to those bursts without relying on overshoot-style quantities, which are difficult to compute robustly under noisy measurements.

Before comparing the different feedback configurations, the experiments followed a simple protocol to ensure repeatable and comparable results.

First, the PBVS loop was tested from several different initial robot poses relative to the same fiducial marker pose. The goal was to confirm that the system converged reliably across the workspace and that the behaviour was not tied to a particular starting configuration.

Second, for a fair controller comparison, all quantitative results in this section were obtained from one specific reference case: a fixed initial robot offset with respect to the marker. This same starting configuration was repeated across runs while changing only the controller gains ( $K_p$ ,  $K_d$ ) and the visual feedback mode (hybrid predictive Kalman or continuous Kalman-filtered). This ensured that differences in  $T_s$  and ISE were due to the controller and the feedback strategy, not to different initial conditions.

Across all runs, the fiducial marker remained stationary, and camera calibration, lighting conditions, convergence thresholds, and control frequency were kept constant to isolate the effect of the controller and filtering strategy.

The inclusion of an integral term in the visual servoing controller was intentionally avoided. The integral term is used to remove steady-state error in classical PID control. However, in a PBVS loop driven by visual measurements this logic does not directly apply, for three reasons:

1. **The task demands convergence within a tolerance, not absolute zero error:** Position and orientation convergence thresholds are explicitly defined by the controller. Once the robot is within those tolerances, the task is considered successful. There is no practical benefit in forcing the residual error to reach zero.
2. **The visual error is noisy and biased:** Pixel quantization, lens calibration error, illumination variations, and partial occlusion can all affect the *ArUco*-based pose estimate. This implies that even when the robot is properly aligned, the measured error never reaches precisely zero. An integral term would keep integrating residual noise, which promotes slow limit cycles and windup rather than improving alignment.
3. **The final approach may become unstable due to the integral term:** The controller operates at extremely low speeds and in close proximity to physical structures close to the target. As the robot is attempting to settle, integral accumulation during the transient can generate an excessively large correction command at the worst possible moment.

In conclusion, the integral action in this application is unnecessary and counterproductive. The robot is already controlled within the required angular and positional tolerances by proportional and PD controllers.

To keep the discussion interpretable and aligned with the control objectives, the figures show a small set of representative controllers. The goal is to highlight the influence of  $K_p$  and  $K_d$  on  $T_s$  and ISE, using the same representative set reported in the corresponding tables.

### 4.3.1 Hybrid predictive–Kalman feedback.

In this configuration, the PBVS loop employs the previously described hybrid visual feedback, combining direct *ArUco* detections with short-horizon Kalman predictions. This approach enhances robustness to temporary occlusions while maintaining low latency in the control loop.

Figure 4.9 shows the evolution of the signed errors in  $X$ ,  $Y$ , and yaw under hybrid feedback for the representative controllers. Since the mission objective is to reach the acceptance range quickly and without accumulating large transient error, the evaluation is based on  $T_s$  and ISE.

Figure 4.10 compares the integrated squared error for these representative tunings. The numerical results are summarised in Table 4.3. The chosen controller within this representative set is the one that produces the lowest cumulative squared error and enters the tolerance band more quickly. The PD variants provide more damping in some intervals, but in these runs they do not improve the global ISE nor the first entry time into tolerance.

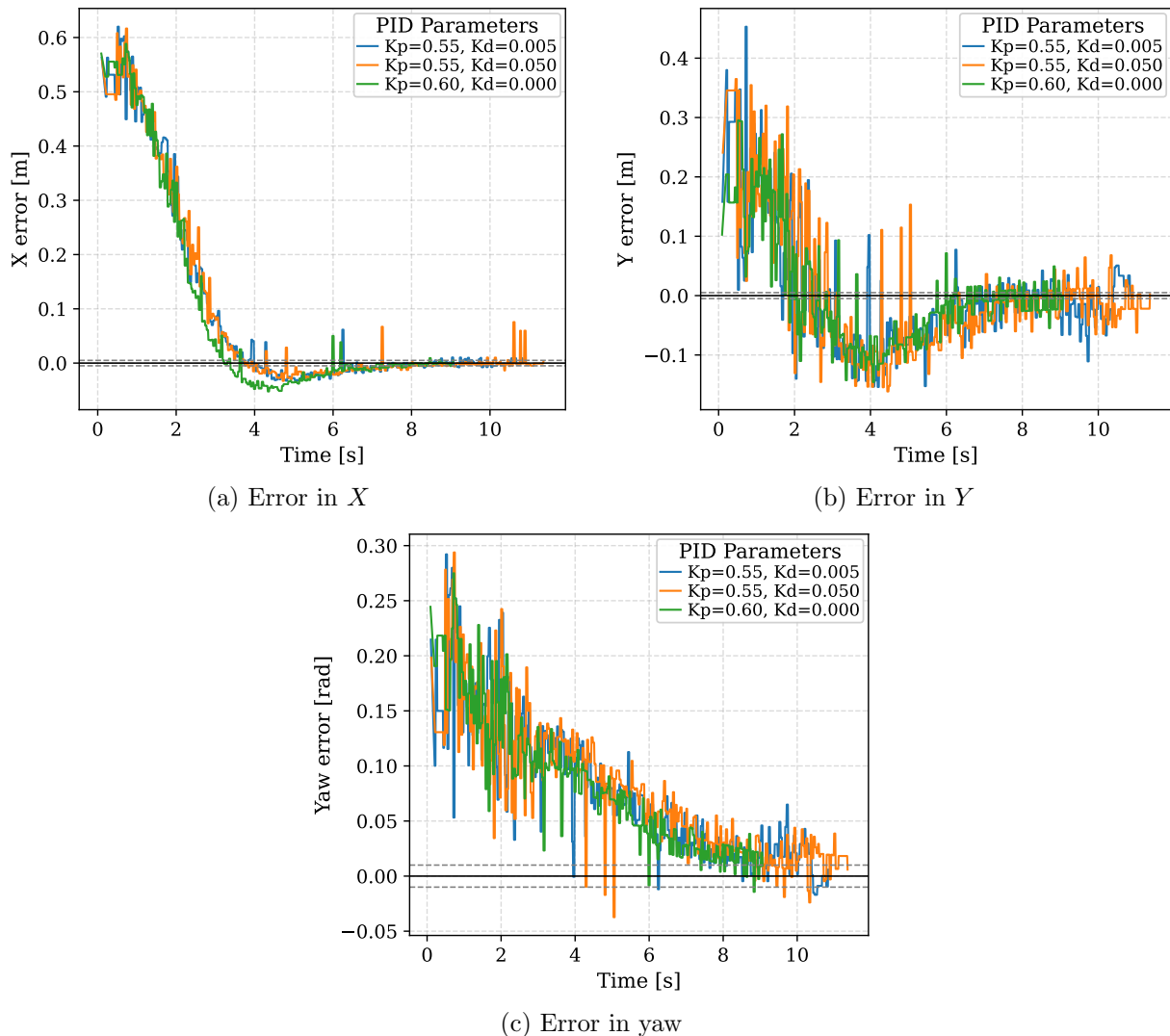


Figure 4.9: Errors in  $X$ ,  $Y$ , and yaw under the hybrid predictive-Kalman feedback for representative controllers.

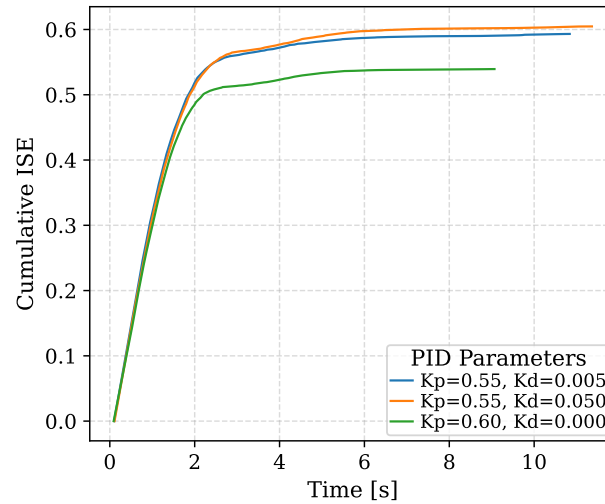


Figure 4.10: Integrated squared error (ISE) for representative controllers under hybrid predictive–Kalman feedback.

Table 4.3: Performance under hybrid predictive–Kalman feedback. Metrics reported as time to enter tolerance ( $T_s$ ) and average integrated squared error (ISE) across  $X$ ,  $Y$ , and yaw.

| Controller          | $K_p$       | $K_d$        | $T_s$ [s]   | ISE           | Remarks                          |
|---------------------|-------------|--------------|-------------|---------------|----------------------------------|
| P (baseline)        | 0.40        | 0.000        | 12.79       | 0.2629        | Slow, large accumulated error    |
| P (moderate)        | 0.70        | 0.000        | 13.81       | 0.2160        | Lower ISE, but slower entry      |
| <b>P (selected)</b> | <b>0.60</b> | <b>0.000</b> | <b>9.07</b> | <b>0.2137</b> | <b>Fast entry with low ISE</b>   |
| PD (low $K_d$ )     | 0.55        | 0.005        | 10.84       | 0.2330        | Damping increases, higher ISE    |
| PD (high $K_d$ )    | 0.55        | 0.050        | 11.36       | 0.2379        | Highest ISE among shown PD cases |

#### 4.3.2 Continuous Kalman-filtered feedback.

In the second set of experiments, the PBVS loop used continuous Kalman-filtered feedback, where all pose estimates were merged over time based on statistical consistency. This approach trades reactivity for more stable behaviour by filtering frame-to-frame noise of raw detections. As a result, the controller commands are less affected by instantaneous pose jitter, which is especially important in the final approach where small oscillations can delay entry into the tolerance band and increase accumulated error.

Figures 4.11 and 4.12 show the error traces and the corresponding ISE comparison for representative controllers. The quantitative results are summarised in Table 4.4. Increasing the proportional gain shortens the time to enter tolerance, while a moderate derivative term reduces the accumulation of squared error by damping residual jitter-driven corrections.

Within the representative set, the selected controller is the one that yields the lowest ISE while keeping  $T_s$  comparable to the fastest proportional tuning. This provides the best overall balance between entering the acceptance band quickly and avoiding persistent squared-error accumulation due to measurement noise.



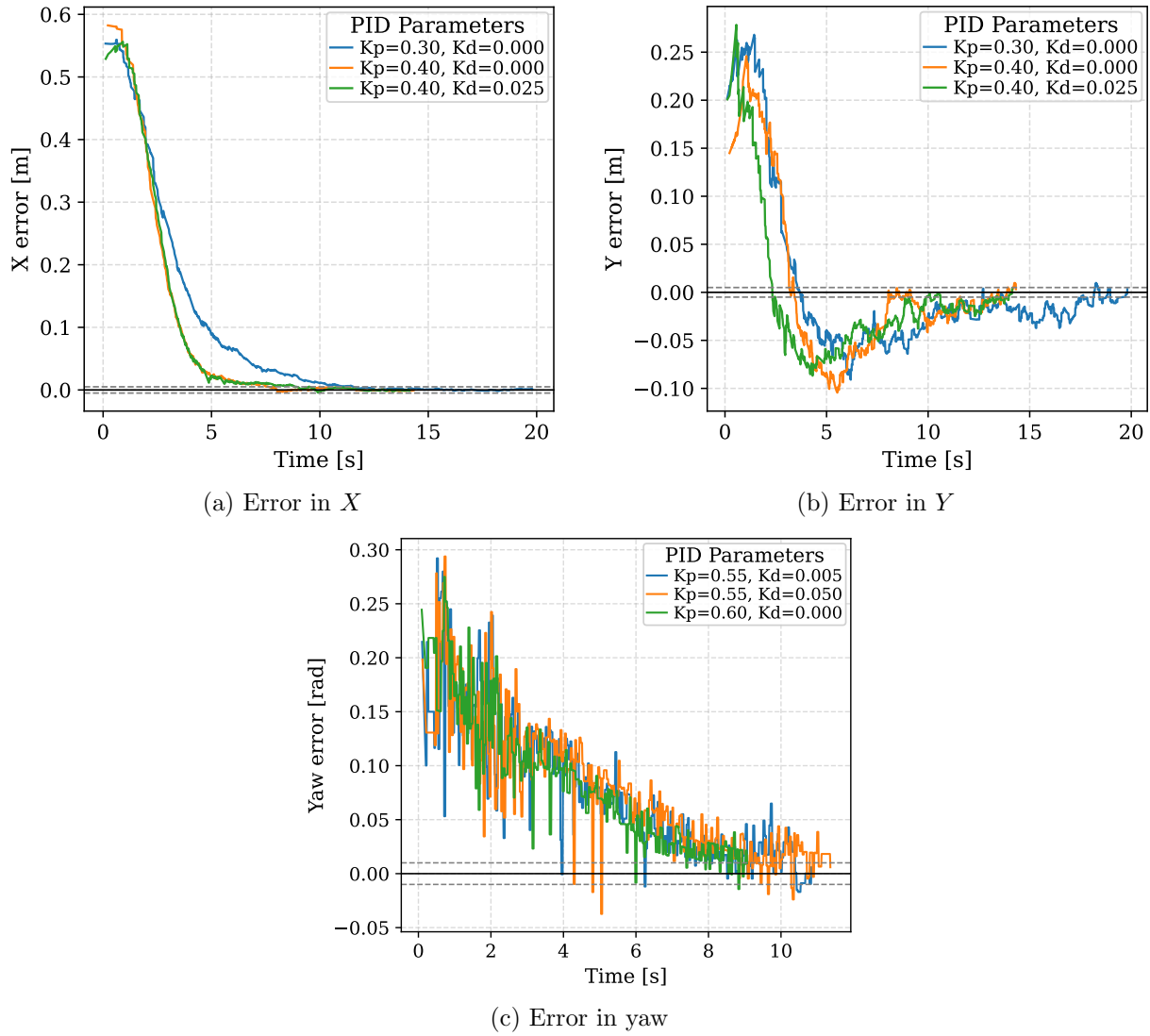


Figure 4.11: Errors in  $X$ ,  $Y$ , and yaw under continuous Kalman-filtered feedback for representative controllers.

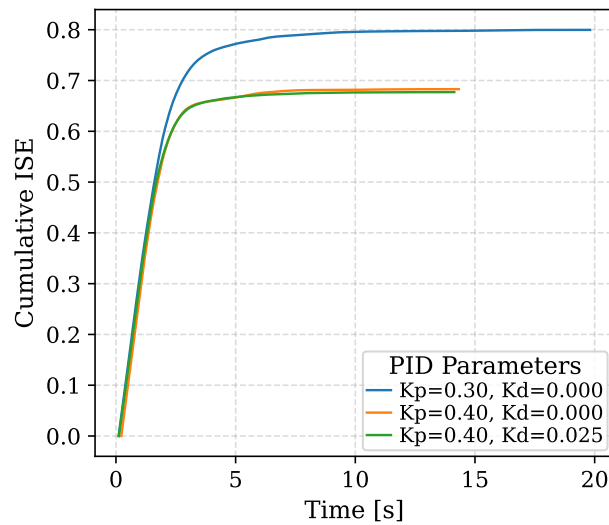


Figure 4.12: Integrated squared error (ISE) for representative controllers under Kalman-filtered feedback.

Table 4.4: Performance of P and PD controllers under continuous Kalman-filtered visual feedback. Metrics reported as time to enter tolerance ( $T_s$ ) and integrated squared error (ISE), averaged across  $X$ ,  $Y$ , and yaw.

| Controller           | $K_p$       | $K_d$        | $T_s$ [s]    | ISE          | Remarks                                             |
|----------------------|-------------|--------------|--------------|--------------|-----------------------------------------------------|
| P (slow)             | 0.30        | 0.000        | 19.81        | 0.352        | Very smooth, slow entry                             |
| P (balanced)         | 0.40        | 0.000        | 14.32        | 0.290        | Faster entry, higher ISE                            |
| P (high $K_p$ )      | 0.45        | 0.000        | 13.03        | 0.286        | Faster, increased squared error                     |
| P (very high $K_p$ ) | 0.50        | 0.000        | 16.71        | 0.321        | Slow and inaccurate in this run                     |
| PD (low $K_d$ )      | 0.40        | 0.010        | 14.51        | 0.285        | Reduced ISE vs P, slightly slower                   |
| PD (medium $K_d$ )   | 0.40        | 0.015        | 14.96        | 0.290        | Similar ISE, no clear benefit                       |
| <b>PD (selected)</b> | <b>0.40</b> | <b>0.025</b> | <b>14.13</b> | <b>0.269</b> | <b>Lowest ISE with competitive <math>T_s</math></b> |
| PD (high $K_d$ )     | 0.40        | 0.030        | 14.59        | 0.274        | Increased sensitivity, higher ISE                   |

#### 4.3.2.1 Final controller choice for deployment.

The controller selection in this study follows the mission success requirement, reaching the acceptability range quickly while minimising the accumulated squared error under noisy visual feedback. For that reason, the comparison is based on  $T_s$  and ISE.

Under hybrid predictive–Kalman feedback, the selected proportional tuning ( $K_p = 0.60$ ,  $K_d = 0.00$ ) provides fast entry into tolerance with low ISE among the reported configurations, and remains a valid option when low-latency reactivity is prioritised.

However, for deployment in close proximity to physical structures, smooth and predictable motion during the final approach is more important than marginal differences in convergence speed. Continuous Kalman-filtered feedback produces cleaner trajectories by attenuating frame-to-frame pose jitter, reducing high-frequency command variations and mechanical stress on joints and transmissions, which improves repeatability and safety.

Therefore, the controller selected for deployment is the PD regulator under continuous Kalman-filtered feedback with  $K_p = 0.40$  and  $K_d = 0.025$  (Table 4.4). This tuning achieves the lowest ISE while keeping a competitive  $T_s$ , providing the best balance between convergence speed and smooth closed-loop behaviour. Although the quantitative comparison is reported for one reference initial offset, additional trials from multiple initial poses within the explored workspace showed consistent convergence, supporting the generality of this choice over the tested range.

In conclusion, the created visual servoing subsystem offers a reliable and reusable control element that allows the robot to independently align with visual objects in difficult situations. Higher-level duties like gate crossing and charging station alignment, which are covered in the next chapter, may be executed precisely thanks to its interface with the mission manager.

## Chapter 5. Mission Manager for Autonomous Radiation Surveys

Building upon the framework described in Chapter 3, this chapter presents the implementation of the Mission Manager developed for the SPS autonomous radiation survey mission. It describes its hierarchical FSM structure, operational flow, and the software mechanisms for safe, deterministic, and supervised execution.

During radiation surveys, the Mission Manager is in charge of coordinating the robot's autonomous behavior. It combines gate crossing, charging, navigation, and radiation logging under a single hierarchical FSM that is implemented in *Boost SML*. This design ensures traceable mission flow, fault-tolerant execution, and clear operator supervision.

### 5.1 Mission Flow and Operational Logic

Understanding the general mission flow that controls the system's autonomous behavior is crucial before going into detail about the software implementation. In Figure 5.1, the robot's whole operational process during a routine radiation inspection inside the SPS tunnel is illustrated.

The mission starts once the operator triggers a **Start** command from the control interface. The Mission Manager transitions through an initialization phase (**INIT**), where system readiness checks are performed, and then enters an idle state (**IDLE**) awaiting mission execution. Before starting autonomous motion, the system evaluates whether the robot is currently located inside the charging station (**IN\_CHARGING\_STATION**). If the robot is docked, the Mission Manager first executes **EXITING\_CHARGING\_STATION** to safely leave the docking area and reach a configuration suitable for motion. Otherwise, the system transitions directly to **NAVIGATION**.

#### Supervised execution policy

Although the Mission Manager evaluates mission conditions continuously, mission progression is supervised by design. Guards determine when a transition becomes eligible, but its commitment is operator-gated. In practice, the supervisory interface displays the proposed step, and the operator validates it by issuing **Next**. Throughout this chapter, this mechanism is referred to as *supervised execution*.

Once in **NAVIGATION**, the Mission Manager calls `activateAutopilot()`. From this point, the low level navigation and motion execution are handled by the Autopilot module, while the Mission Manager remains responsible for high level supervision, mode switching, and safety relevant decisions. The Autopilot module is in charge of generating and executing motion commands, including local replanning and path tracking according to its internal configuration. In contrast, the Mission Manager supervises mission level conditions and triggers transitions to specific procedures, such as gate crossing or docking, under supervised execution.

If the system detects a gate along the route, the corresponding guard becomes true and control is sent to the **GateCrossing** sub-FSM. This transition marks the entry into a constrained procedure that encapsulates the full safety-gate traversal behaviour, including precision alignment, crossing, and the corresponding safety checks. Once the sub-FSM completes successfully, it returns an exit outcome to the main **System** FSM, which resumes normal navigation with Autopilot enabled. If the procedure is aborted or an error condition is detected, the main FSM transitions to the **ERROR\_HANDLING** sub-FSM.

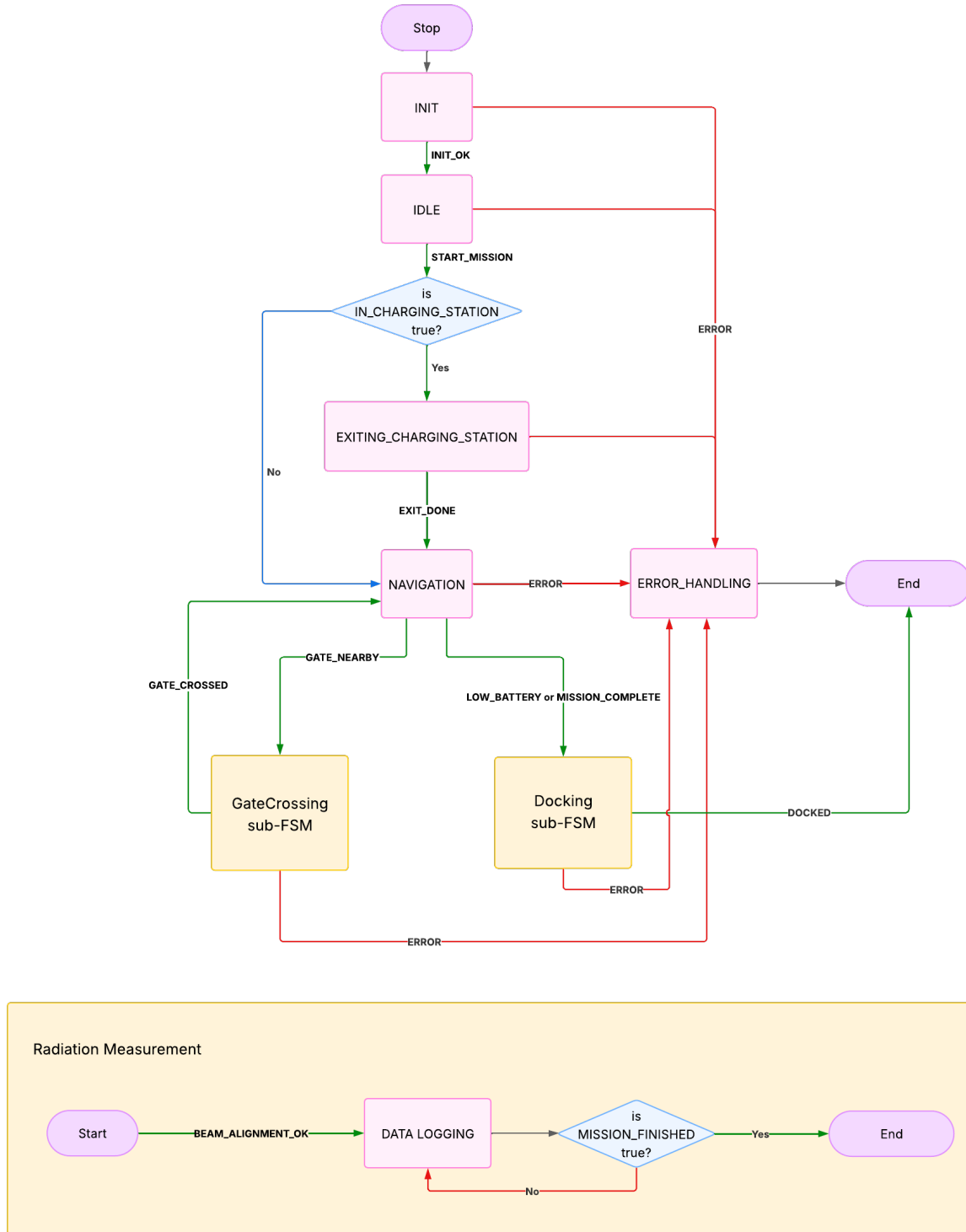


Figure 5.1: Mission flowchart representing the high-level operational logic of the autonomous survey procedure.

At any moment, if a docking request is triggered due to energy constraints or if a mission-completed condition is raised, the Mission Manager activates the **Docking** sub-FSM. This transition starts the specialized return and docking procedure, which manages the complete charging-station approach and docking workflow under supervised execution. Once docking is completed successfully, the sub-FSM reports a **DOCKED** outcome, and the mission is considered finished,

leading the main FSM to a terminal safe state. If docking fails or an error condition is detected, the sub-FSM reports its outcome to the parent FSM, which transitions to `ERROR_HANDLING` for recovery or operator fallback.

In this design, radiation measurement runs concurrently as a dedicated process, represented in Figure 5.1 as an independent *Radiation Measurement* flow. This parallel logic is responsible for continuous data logging once beam alignment is confirmed. When the condition `BEAM_ALIGNMENT_OK` becomes true, the radiation process enters the `DATA LOGGING` state and records dose rate measurements continuously. The measurement process continues to log until `MISSION_FINISHED` is claimed, at which time it ends deterministically. While maintaining continuous radiation monitoring during the survey, this decoupling guarantees that navigation, gate crossing, and docking stay independent from the sampling approach.

The flowchart shows how each mission phase is connected through conditional transitions and safety checks. These decision branches, such as whether the robot starts docked, whether a gate is detected, whether docking is required, or whether an error occurs, are directly mapped to the guards implemented in the software FSM. As a result, the chart works as a functional overview corresponding to the internal state changes managed by the `RPSurveySPS` mission manager.

This structured approach ensures that all possible situations, such as normal operation, environmental interactions, and safety triggers, are explicitly modeled. By combining navigation supervision, gate crossing, and docking behaviours into a coherent flow, and by running radiation logging concurrently, the mission manager achieves autonomous survey execution while maintaining deterministic control and recovery mechanisms in the event of a failure.

### 5.2 Software Structure and Execution Flow

The `RPSurveySPS` mission manager integrates the hierarchical FSM architecture described previously into a complete and fully functional software framework written in modern C++ and based on `Boost.SML`. Its design follows a modular approach that separates decision logic, event definition, guard conditions, and motion-level actions, allowing the system to remain clear, maintainable and easily extensible.

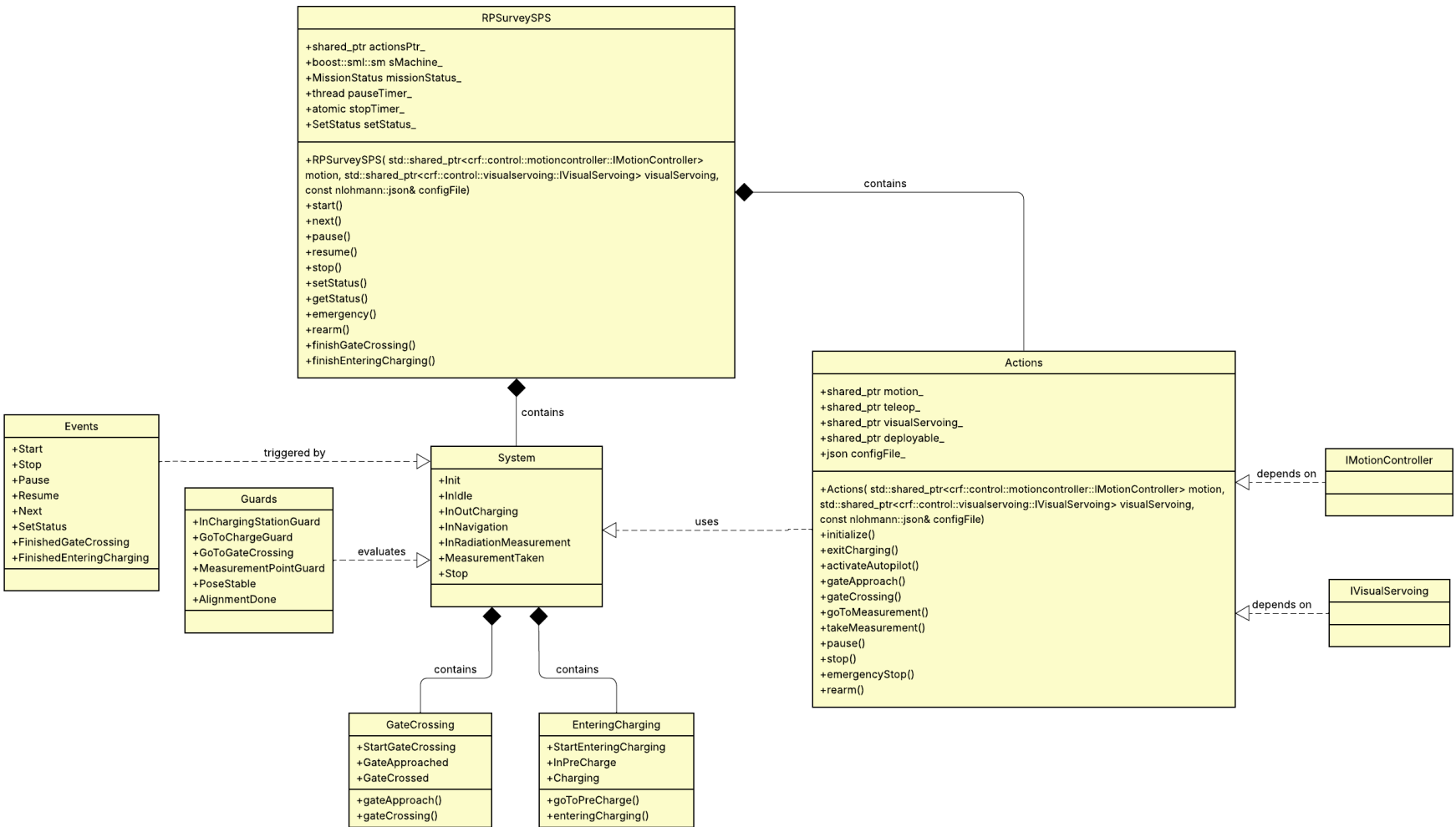


Figure 5.2: UML class diagram of the RPSurveySPS mission manager and its core components.

As shown in Fig. 5.2, the implementation is structured around four key elements: events, guards, actions and finite state machines. Events represent the discrete triggers that drive transitions between states. They are defined in `Events.hpp` and include mission-level commands such as `Start`, `Pause`, `Resume` or `Stop`, as well as two key events used in supervised execution: `SetStatus` and `Next`. The `SetStatus` event updates the internal mission context from external status messages, for instance `GateNearby`, `MeasurementPointReached` or `LowBattery`. Guards evaluate this context to decide which transitions are eligible. The `Next` event commits the selected transition once the operator validates the proposed step in the supervisory interface.

Guards act as boolean conditions that evaluate whether a transition should occur. Implemented in `Guards.hpp`, they provide real-time feedback through console messages, making the execution traceable during testing and debugging. Typical examples include `GoToChargeGuard`, which activates the charging sequence when the battery level is low; `GoToGateCrossing`, which triggers the gate-crossing routine when a gate is detected; and `MeasurementPointGuard`, which starts the measurement routine when a predefined waypoint is reached.

Actions define the actual robot operations executed when a transition takes place. They encapsulate all interactions with the CRF motion and vision controllers and are implemented in `Actions.cpp`. This class provides functions such as `exitCharging()`, `goToPreCharge()`, `enteringCharging()`, `activateAutopilot()`, `gateApproach()`, `gateCrossing()` and `takeMeasurement()`. Each action communicates with hardware-level interfaces like `IMotionController` and `IVisualServoing`, ensuring a clean separation between decision logic and low-level control.

The state machines themselves are defined in the classes `System`, `GateCrossing` and `EnteringCharging`. The `System` FSM represents the main mission logic, while the other two encapsulate specialised procedures for safety-gate traversal and charging-station docking. In the final architecture, constrained procedures are executed exclusively, meaning that only one sub-FSM can be active at a time. This is achieved through a dedicated procedure container at the parent level, with a single entry transition into each sub-FSM and a single exit event returned to the `System` FSM. This layout keeps procedure logic self-contained, reduces coupling between the parent FSM and sub-FSM internal states, and improves maintainability and traceability.

At runtime, external modules interact with the mission manager through the `IMissionManager` API. High-level functions such as `start()`, `next()`, `pause()` or `stop()` generate events that are forwarded to the internal state machine using:

```
sMachine_.process_event(EventName{});
```

When an event is received, the FSM checks the necessary guards, carries out the related actions, and, if all requirements are satisfied, completes the transition. In supervised execution, guards may indicate that one or more transitions are eligible, but progression occurs only when `Next` is issued. This event-driven structure keeps the logic easy to understand and guarantees deterministic execution.

The method `setStatus()` acts as the main communication bridge between the mission manager and higher-level layers such as the CRF GUI or external control scripts. It receives a JSON message containing the current mission status (for instance `LowBattery`, `GateNearby` or `MeasurementPointReached`), maps these values to internal flags and then raises a `SetStatus` event. This mechanism updates the mission context and allows the FSM to determine the next eligible transition, while the commitment remains controlled through `Next`.

### 5.3 Error Handling, Recovery and Operator Supervision

Radiation survey missions at CERN must remain supervised during operation, both in real deployments and in laboratory tests. This requirement is procedural. The Mission Manager provides autonomy for navigation and procedures, but the operational workflow is executed under supervised execution, ensuring that critical steps and recovery actions remain auditable and intentional.

To put supervision into practice, condition evaluation and transition commitment are kept apart. System and sensor conditions are provided through `setStatus()`, making guards true or false, while the transition itself is only committed when `Next` is issued via the supervisory interface.

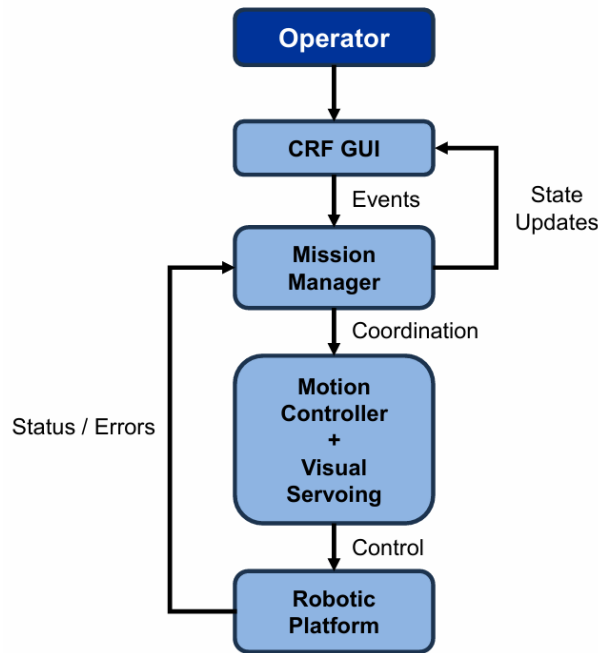


Figure 5.3: Operator-in-the-loop architecture used in real and simulated execution. The FSM evaluates conditions autonomously, while progression is operator-gated through `Next`.



## Error Handling Architecture

Error handling in the Mission Manager follows a layered strategy. First, each mission sub-FSM performs local checks and exception handling for failures such as motion timeouts, marker loss, or inconsistent sensor readings. When a failure cannot be resolved locally, it is escalated to the parent **System** FSM and mapped to a global error event. To guarantee a deterministic and auditable response, global recovery is implemented as a dedicated **ErrorHandling** sub-FSM, shown in Figure 5.4.

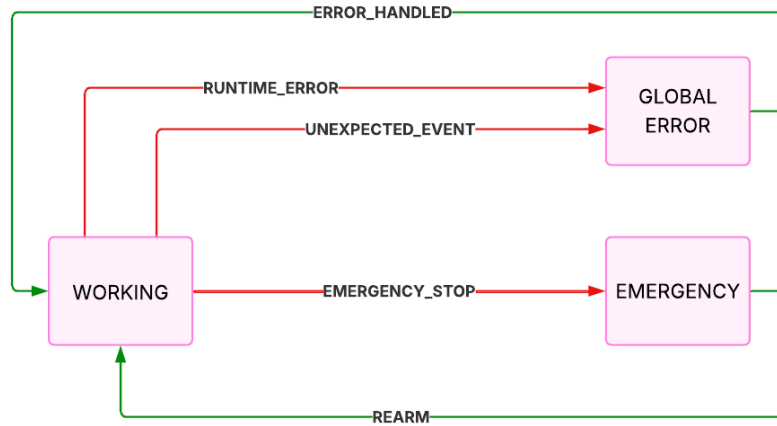


Figure 5.4: State machine diagram for the global error handling procedure. Recoverable failures transition from **WORKING** to **GLOBAL ERROR**, while safety-critical stops transition to **EMERGENCY**.

The **ErrorHandling** sub-FSM defines three explicit modes:

- **WORKING:** Normal operation. The mission FSM executes the current procedure under supervised execution.
- **GLOBAL ERROR:** Recoverable fault state entered upon **RUNTIME\_ERROR** or **UNEXPECTED\_EVENT**. In this state, the system remains in a controlled configuration, logs the fault, and awaits an explicit recovery decision. Once the condition is resolved, the **ERROR\_HANDLED** event returns the system to **WORKING**.
- **EMERGENCY:** Safety-critical stop state entered upon **EMERGENCY\_STOP**. Actuation is disabled and motion is halted immediately. Recovery from this state is never autonomous. A manual **REARM** command, executed only after verification and hardware-level clearance, returns the system to **WORKING**.

This structure distinguishes emergency stops from recoverable runtime faults and ensures that returning to operational mode always requires an intentional recovery action, either **ERROR\_HANDLED** or **REARM**.

## Recovery Logic

Table 5.1 summarises the recovery actions applied by the Mission Manager for gate crossing and docking related manoeuvres. Local failures are handled inside the corresponding sub-FSM when possible. If intervention or a system restart is required, the failure is escalated to the global error handling layer, and the appropriate recovery action is selected.

| Recovery logic used by the mission manager   |                                                                        |
|----------------------------------------------|------------------------------------------------------------------------|
| Failure                                      | Recovery Strategy                                                      |
| Move to arm configuration failed             | Repeat the action                                                      |
| Emergency stop triggered                     | Restart the system                                                     |
| Collision with the wall during charging exit | Teleoperate the robot or repeat the action                             |
| Marker not detected or lost                  | Teleoperate the robot or repeat the action                             |
| Persistent oscillation in visual servoing    | Repeat the action or restart with adjusted tolerance or PID parameters |
| Driver failure                               | Restart the system                                                     |

Table 5.1: Recovery logic used by the mission manager.

## Operator Role during Recovery

For recoverable errors, the operator decides whether the action can be safely repeated, whether teleoperation is required, or whether a full restart is mandatory, according to Table 5.1. In the implementation, this decision is reflected as a transition out of **GLOBAL ERROR** through **ERROR\_HANDLED**. For emergency stops, recovery requires explicit clearance of the emergency condition and a **REARM**, ensuring that safety-critical stops cannot be cleared by the autonomous layer.

The combination of automatic safeguards and supervised execution creates a strict, traceable, and certifiable execution environment since the FSM is deterministic and logs every processed event and transition. This dual structure ensures consistent operation and is a crucial requirement for deployment within the SPS tunnel.

## 5.4 Developed Mission Components

This section describes the two constrained mission procedures implemented as dedicated sub-state machines in the `RPSurveySPS` mission manager, namely `GateCrossing` and `EnteringCharging`. They are triggered from the parent `System` FSM when their guards become true and are executed under the supervised execution policy through operator-gated `Next`. Both behaviours rely on *ArUco*-based PBVS to achieve repeatable alignment before performing short, highly constrained motions in the SPS tunnel, and they follow the recovery logic summarised in Table 5.1.

### 5.4.1 Gate Approaching and Gate Crossing

This procedure is executed by the `GateCrossing` sub-state machine of the `RPSurveySPS` mission manager. The objective is to cross the gate in a repeatable way by performing a precise PBVS alignment and then running a straight crossing motion.

Accurate alignment is critical because the gate is crossed through a narrow cut-out. The admissible lateral and yaw error is therefore small, and a residual pose error at the start of the traversal can lead to collision with the frame or an unsuccessful crossing. For that reason, the robot only initiates the crossing once the visual alignment has converged to a sufficiently low residual error.

#### Gate Marker Placement and PBVS Target

The gate marker used for alignment is an *ArUco* tag with `MarkerID=0` and side length of 8.8 cm. It is placed immediately above the cut-out and centered with respect to the opening, as shown in Figure 5.5. This placement maximizes the probability that the marker remains in the field of view of the front camera during the approach and, by construction, enforces an alignment consistent with the geometry of the cut-out.

The marker side length (8.8 cm) was selected as a compromise between detection robustness and installation constraints. A larger tag increases the apparent size in the image, improving detection stability and pose estimation accuracy at the approach distance, while excessively large tags become harder to place, more prone to partial occlusion, and less representative of the local gate geometry. The chosen size provides a sufficiently strong visual signal at the working range while remaining easy to mount above the opening and repeatably printable without distortion.

The PBVS target pose for `GateApproach` is defined in the marker frame at a nominal distance of 0.8 m along the marker normal (camera optical axis at convergence), with the corresponding target rotation specified in the configuration. This target distance was selected as the closest range at which the marker remains reliably within the front camera field of view throughout the final correction, enabling the most accurate alignment immediately before initiating the straight traversal.



Figure 5.5: Gate marker placement.

### Visual Alignment and Crossing

The routine starts once the robot is located at the predefined *Survey Position*. From this pose, a PBVS alignment is executed using the front camera until convergence to the **GateApproach** target (Figure 5.6a). After convergence, the robot transitions the arm from the survey configuration to the crossing configuration and commands a straight 2.5 m motion along the mission frame  $x$  axis (tunnel longitudinal direction) (Figure 5.6b). Once the traversal is completed, the arm returns to the survey configuration and the mission continues.

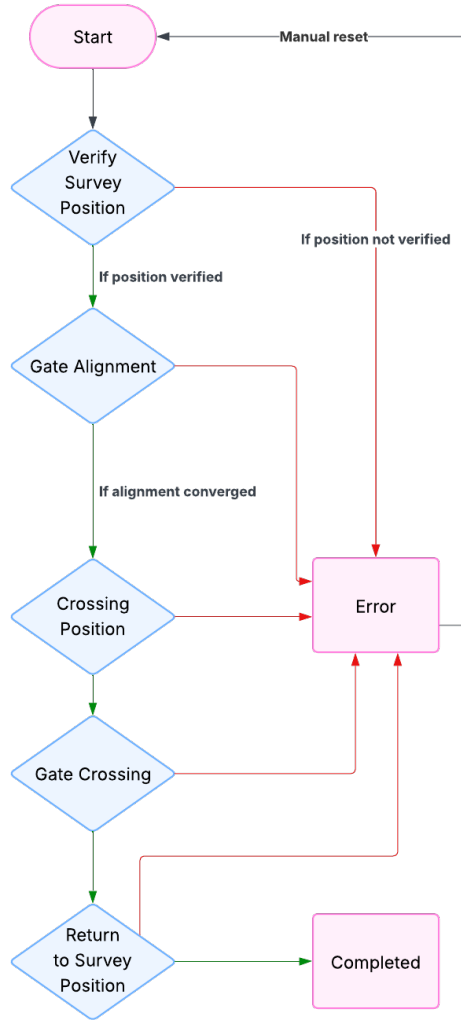


(a) Alignment.



(b) Crossing.

Figure 5.6: Gate crossing sequence.

Figure 5.7: State machine diagram for the **GateCrossing** procedure.

#### 5.4.2 Parking Procedure and Docking into the Charging Station

This routine is executed by the **EnteringCharging** sub-state machine of the **RPSurveySPS** mission manager. Its goal is to autonomously park inside the charging station and complete the electrical docking by combining three visual servoing alignments with intermediate repositioning maneuvers adapted to SPS tunnel constraints.

In the SPS tunnel, charging stations are installed next to a left-side wall, which constrains the approach corridor and makes right-side approaches the most common operational scenario (Figure 5.8). Consequently, the initial visual alignment must be feasible from a pose that is laterally offset to the right, with the robot still outside the station and the arm in survey configuration.

For this reason, the first alignment marker (**PreCharge**) is placed on the upper-right corner of the station entrance (Figure 5.9). This location maximizes visibility from the front camera during the right-side approach and reduces the risk of occlusions caused by the left wall or by the station frame, producing a repeatable entrance pose aligned with the ramp corridor.

The station includes an access ramp. This ramp is required because water leaks can occur in the SPS tunnel, therefore the station is elevated to avoid exposing sensitive mechanical and electrical elements at floor level. However, ramp traversal introduces additional uncertainty. With mecanum wheels, lateral slip and small yaw deviations can happen while ascending. In this context, visual servoing is essential because it provides a continuous visual reference to compensate drift and reduce the risk of colliding with the station side walls or failing the ramp traversal due to slip-induced deviation.



Figure 5.8: Charging station layout in the SPS tunnel.

### Charging Station Marker Placement and PBVS Targets

Two *ArUco* markers are used (Figure 5.9). The **PreCharge** marker (**MarkerID**=20, 8.8 cm) supports the first alignment using the front camera. During the entry approach, when the robot may still be at a relatively small distance and minor heading mistakes can cause the markers's apparent size to decrease, its size matches the gate marker to ensure comparable detection robustness.

The **FinalCharging** marker (**MarkerID**=2, 8.0 cm), placed inside the station at the back, is used for the two rear-camera alignments required to enter the station in reverse and to achieve an accurate connection. A slightly smaller *ArUco* is enough in this stage because the robot operates at shorter range inside the station, and it better fits the available mounting area without interfering with mechanical elements or increasing the risk of occlusions during reverse insertion.

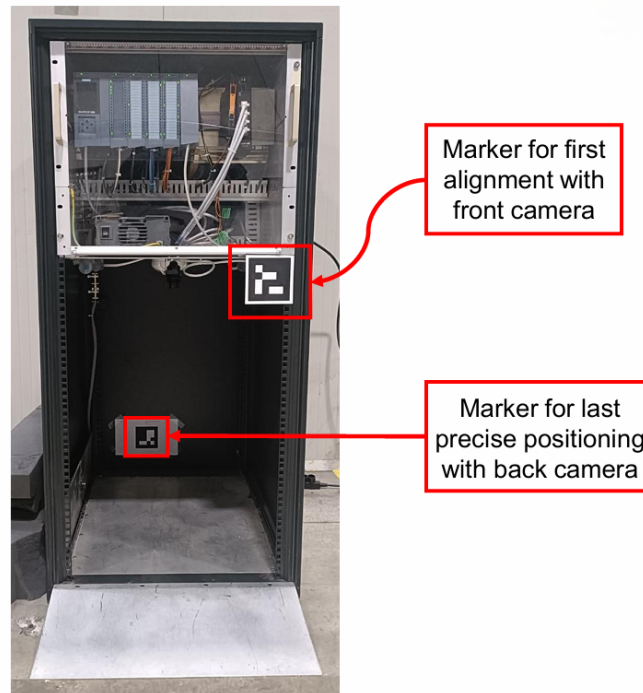


Figure 5.9: *ArUco* marker placement on the charging station.

## Parking Sequence

The procedure starts at the predefined *Survey Position*.

1. **Front alignment at the station entrance:** The robot performs a first PBVS alignment using the **PreCharge** marker with the front camera and the arm in survey configuration (Figure 5.10a). This stage gives a repeatable pose in front of the station, aligned with the entrance under the typical constraints imposed by the mission.

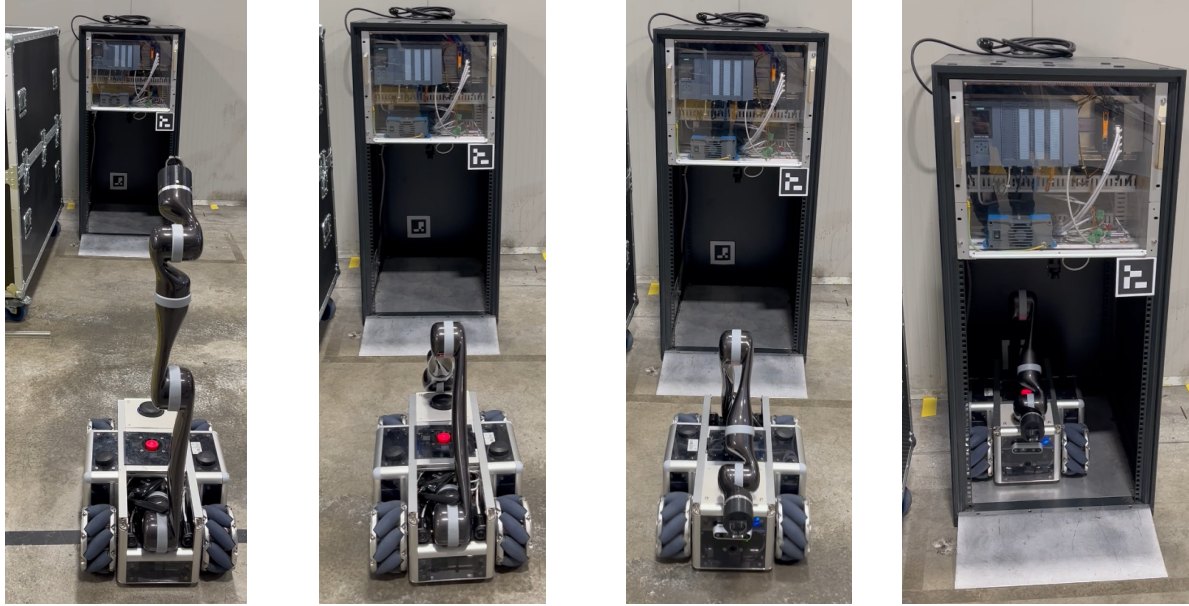
After the first alignment, the robot transitions the arm to the charging configuration and advances straight towards a staging pose near the ramp (Figure 5.10b). The staging distance is selected to be close enough to keep a controlled approach, but not too close, so that the robot can accelerate over a short distance and achieve enough speed to climb the ramp reliably.

The robot then rotates  $180^\circ$  in place to switch perception to a rear camera (Figure 5.10c), preparing reverse insertion.

2. **Rear alignment for reverse entry:** Using the rear camera, the robot performs a second PBVS alignment with the **FinalCharging** marker located inside the station. This stage is used to position the robot in front of the station while already oriented for reverse motion, minimizing lateral and yaw error before starting the reverse insertion.
3. **Rear alignment for precise docking and connector engagement:** After the reverse entry begins, the robot runs a third visual servoing stage, again using the rear camera and the **FinalCharging** marker. This final stage maintains the marker reference while the robot moves deeper into the station and performs the precise positioning required to correctly plug into the lateral charging connector (Figure 5.10d). Keeping the visual



reference during this last phase is particularly important to compensate ramp-induced slip and avoid contact with the station side walls.



(a) Front PBVS with (b) Approach to ramp (c) 180° rotation to rear (d) Rear PBVS with  
PreCharge. staging pose. camera. FinalCharging.

Figure 5.10: Parking and docking sequence, front alignment, ramp staging, switch to rear perception, and reverse alignment plus final docking.

## Error Handling and Operator Control

For **GateCrossing** and **EnteringCharging**, failures such as marker loss, configuration transitions that do not complete, or unstable PBVS behaviour are handled according to Table 5.1. Marker not detected or lost conditions trigger either a repetition of the action or operator teleoperation, depending on safety conditions and mission context. Persistent oscillations during visual servoing are treated as a control instability condition, therefore the action is repeated, or the system is restarted after updating pose tolerances or PID gains. Emergency stop events and driver failures trigger a mandatory system restart. In simulation mode, operator decisions are validated through **SetStatus**, enabling systematic testing of the same recovery logic without relying on real sensor feedback.



## Chapter 6. Experimental Validation and Performance Results

This section reports the experimental validation and performance evaluation of the integrated system at two complementary levels.

At procedure level, the gate crossing and charging station parking procedures described in Sections 5.4.1 and 5.4.2 are validated through repeated trials. In these experiments, the Mission Manager contributes by enforcing a deterministic execution structure. It selects the appropriate procedure based on mission context, triggers the corresponding actions in a consistent order, and applies the same error handling and recovery policy across trials (Section 5.3, Table 5.1). As a result, variability due to operator dependent sequencing is reduced, and repeatability can be assessed under controlled, comparable conditions. Performance is evaluated using execution time, PBVS residual error at convergence, and qualitative trial outcomes.

At mission level, the Mission Manager provides the integration layer that turns local autonomous maneuvers into mission progress. It embeds gate crossing and docking as callable sub-procedures within longer navigation phases, manages their entry and exit conditions, and defines how failures propagate to global recovery. The mission level analysis therefore quantifies not only timing improvements in individual maneuvers, but also how reliable procedure execution and consistent recovery decisions influence the overall mission duration across representative operational modes.

### 6.1 Experimental Protocol and Metrics

Both procedures were evaluated under repeated trials starting from multiple initial robot poses. For each experiment, the robot was placed at different starting positions within a 3 m range relative to either the safety gate or the charging station. This range was selected to capture the typical pose dispersion introduced by autonomous navigation in autopilot mode and to ensure sufficient visual acquisition distance for reliable detection of the corresponding *ArUco* marker before initiating PBVS alignment. From these varying initial conditions, the full procedure was executed 50 times to assess repeatability under realistic variability in approach geometry.

Timing is reported as mean  $\mu$  and standard deviation  $\sigma$  for each sub-step. For PBVS stages, residual translation and rotation errors at convergence are reported using minimum, maximum, and mean values across trials. In addition, trials are classified as *clean* when completed without contact, *touching* when minor contact occurs but the maneuver still completes successfully, and *unsuccessful* when the procedure cannot complete and requires recovery or operator intervention.

During repeated trials, the Mission Manager enforces consistent failure semantics. Recoverable conditions such as marker loss or unstable PBVS behaviour are handled according to the local procedure logic first, and escalated to the global **ErrorHandling** layer only when local resolution fails. This separation ensures that the reported success rates reflect both control performance and the effectiveness of the structured recovery policy described in Table 5.1.

In addition to procedure level metrics, mission level performance is discussed by aggregating procedure timings within a representative mission sequence. This provides context on how short local maneuvers, such as gate crossing and docking, contribute to overall mission duration when interleaved with navigation segments, and how recovery events affect mission level time through repetitions or fallback actions.

## 6.2 Gate Crossing Results

Tables 6.1 and 6.2 report the timing and the PBVS residual error at convergence for the gate approach stage. Across the 50 trials, the procedure achieved a 100% success rate. Most trials crossed cleanly, while a small fraction presented minor contact with the gate lateral frame. These touch events were observed only under very poor illumination conditions, where marker detection quality and pose stability degrade, and they did not prevent successful traversal.

The PBVS alignment stage converged to a mean translation residual of 3.33 mm and a mean rotation residual of  $0.17^\circ$ , providing the precision required before committing to the straight crossing motion through the narrow cut-out. In addition, the Mission Manager contributes to repeatability by ensuring that the same convergence criterion, crossing configuration transition, and forward traversal action are executed in the same order in every trial, and by providing a well-defined recovery path if abnormal conditions are detected.

Figure 6.1 visualizes the time distribution per step, including measured variability. Figure 6.2 summarizes the qualitative outcomes.

| Step                 | Time (s) |          |
|----------------------|----------|----------|
|                      | $\mu$    | $\sigma$ |
| Approach Gate        | 13.17    | 1.78     |
| Arm to Crossing Pose | 8.31     | 0.02     |
| Forward Crossing     | 41.68    | 0.07     |
| Total                | 63.16    | 1.78     |

Table 6.1: Gate crossing timing ( $N = 50$ ).

| Step          | Translation Error (mm) |      |       | Rotation Error ( $^\circ$ ) |      |       |
|---------------|------------------------|------|-------|-----------------------------|------|-------|
|               | min                    | max  | $\mu$ | min                         | max  | $\mu$ |
| Approach Gate | 0.62                   | 4.85 | 3.33  | 0.027                       | 0.27 | 0.17  |

Table 6.2: PBVS residual error at convergence during gate approach ( $N = 50$ ).

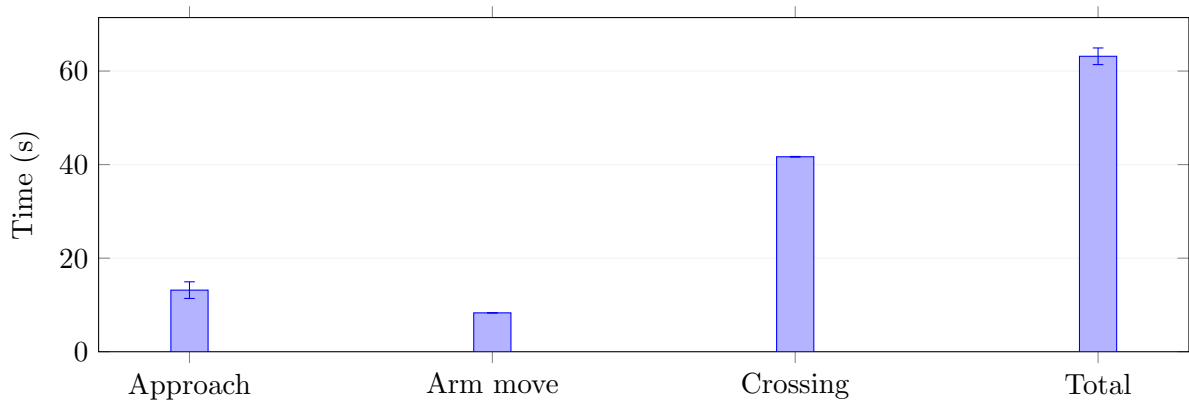
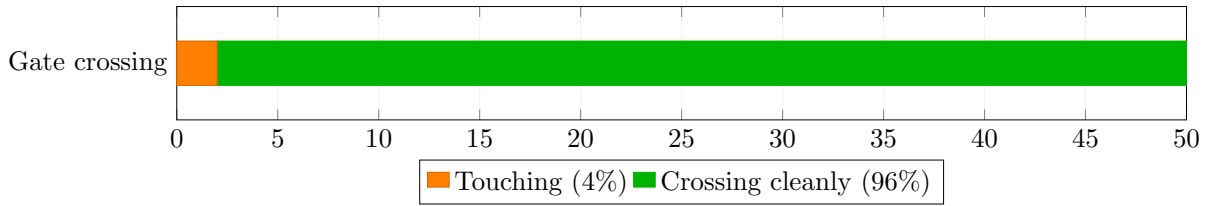


Figure 6.1: Gate crossing execution time per step, mean and standard deviation ( $N = 50$ ).

Figure 6.2: Gate crossing outcomes across trials ( $N = 50$ ).

### 6.3 Parking and Docking Results

Tables 6.3 and 6.4 report the timing and PBVS residual errors for the alignment stages used during the parking procedure. Across the 50 trials, the overall success rate was 98%. Most trials docked cleanly, while a small fraction presented minor contact events with the station structure. One trial was unsuccessful due to a collision with one of the charging station laterals during ramp traversal.

This failure mode was associated with high wheel slip while climbing the ramp using mecanum wheels. In this case, the robot drifted laterally despite the visual reference, resulting in a collision. The observed behaviour indicates a limitation dominated by mechanical design and traction conditions rather than a software failure of the mission manager or the PBVS logic. A mechanical mitigation is currently being investigated by the CERN Robotics Section mechatronic design team to reduce slip on the ramp and improve docking robustness.

From a system perspective, the Mission Manager influences docking performance by standardizing the sequence of alignments and intermediate motions, and by providing a clear recovery pathway when anomalies occur. In particular, conditions such as marker loss, unstable PBVS convergence, or configuration transition failures are mapped to repeat, teleoperation fallback, or restart decisions following Table 5.1. This structured recovery policy limits uncontrolled behaviour and makes failure cases observable and traceable during evaluation.

Figure 6.3 shows the step timing distribution. Figure 6.4 summarizes the qualitative outcomes.

| Step                 | Time (s) |          |
|----------------------|----------|----------|
|                      | $\mu$    | $\sigma$ |
| Pre-Charge Alignment | 16.03    | 13.94    |
| Approach             | 44.02    | 0.04     |
| Final Alignment      | 10.30    | 4.34     |
| Charge Connection    | 6.41     | 0.94     |
| Total                | 76.74    | 13.24    |

Table 6.3: Parking procedure timing ( $N = 50$ ).

| Step                 | Translation Error (mm) |       |       | Rotation Error ( $^{\circ}$ ) |      |       |
|----------------------|------------------------|-------|-------|-------------------------------|------|-------|
|                      | min                    | max   | $\mu$ | min                           | max  | $\mu$ |
| Pre-Charge Alignment | 55.29                  | 73.30 | 63.53 | 0.09                          | 1.18 | 0.65  |
| Final Alignment      | 20.01                  | 55.29 | 27.79 | 0.13                          | 0.81 | 0.33  |
| Charge Connection    | 45.08                  | 48.00 | 46.71 | 0.02                          | 0.27 | 0.16  |

Table 6.4: PBVS residual error at convergence during parking alignment stages ( $N = 50$ ).

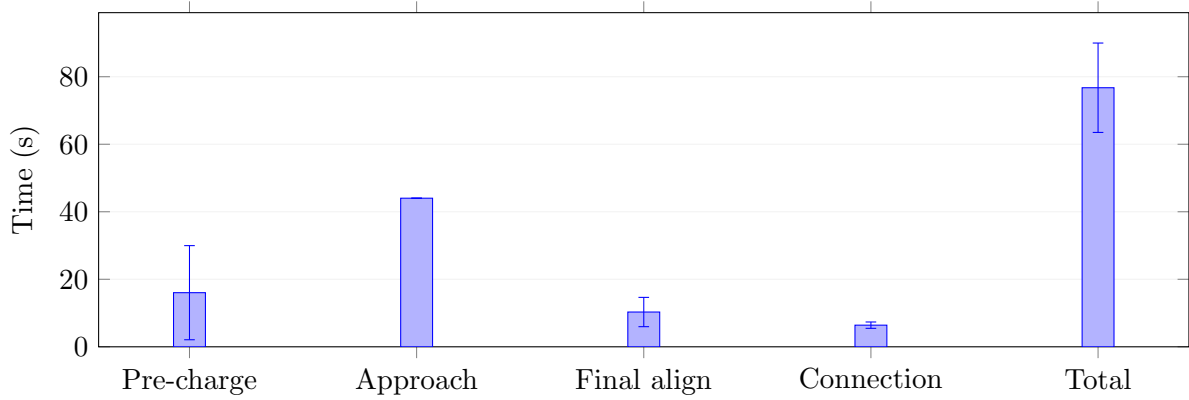


Figure 6.3: Parking procedure execution time per step, mean and standard deviation ( $N = 50$ ).

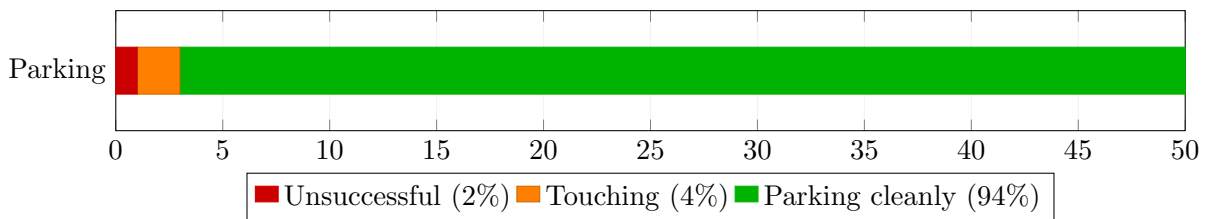


Figure 6.4: Parking procedure outcomes across trials ( $N = 50$ ).

## 6.4 Procedure Level and Mission Level Performance Comparison

To quantify the impact of integrating the mission manager and automated procedures, three operational modes are compared. In this comparison, the Mission Manager is the element that standardizes mission sequencing, procedure invocation, and recovery escalation, enabling a fair mode-to-mode assessment where the main difference is the level of autonomy rather than the order in which steps are executed.

**Teleoperated** refers to full manual intervention during both navigation and local maneuvers.

**Autopilot** refers to autonomous navigation, while local maneuvers such as gate crossing and parking are executed manually as standalone actions, without the mission manager coordinating procedure invocation, sequencing, and recovery transitions.

**Supervised** refers to the fully integrated system, combining autonomous navigation (autopilot), the mission manager, and the automated procedures for gate crossing and parking, with the operator supervising and intervening only when required by the recovery logic.

Procedure level timings are taken from the experimental results reported in Tables 6.1 and 6.3, where both procedures were evaluated in isolation under repeated trials. Mission level performance is then discussed by composing these measured durations within a representative mission sequence, in which gate crossing and docking appear as short, well-defined events embedded in longer navigation phases.

Since a full end-to-end mission could not be executed within the available experimental time, the total mission duration is expressed as an estimate rather than as a directly measured quantity. This estimate is obtained by combining the measured procedure times with navigation and coordination time ranges observed during the same deployment sessions, and by accounting for

the impact of recovery actions using the recovery policy defined in Table 5.1. Navigation remains the dominant contributor to mission duration, while gate crossing and docking contribute as localized events whose timing and reliability are characterized experimentally.

Table 6.5 summarizes both the measured procedure times and the estimated impact at mission level. At procedure level, gate crossing time is reduced from 120 s (teleoperated) to 63.16 s (supervised), corresponding to a 47.37% reduction. Parking time is reduced from 90 s (teleoperated) to 76.74 s (supervised), corresponding to a 14.73% reduction.

At mission level, the benefit is not limited to the time saved during the procedures themselves. By integrating the mission manager, the system reduces coordination overhead between navigation and local maneuvers, and it also limits time losses due to recovery-driven repetitions by enforcing consistent escalation and fallback behaviors. The reported mission duration ranges are obtained by aggregating the measured procedure times with conservative bounds on navigation and coordination time. These bounds reflect variability in starting poses, path length, and replanning events, as well as the observed frequency of recovery actions during procedure level evaluation. As a result, the mission level figures should be interpreted as order-of-magnitude estimates suitable for comparing operational modes under the same assumed mission structure, rather than as guarantees of absolute mission duration.

As a result, the estimated total mission duration decreases from 3.5 h (teleoperated) to 1 to 2 h (supervised), while autopilot mode remains between 1.5 and 2.5 h due to manual execution and manual coordination of local maneuvers.

For clarity, the *Change* reported for the total mission time is expressed as a conservative range spanning the most and least favorable baselines available in the comparison. The lower bound,  $-20\%$ , corresponds to the most conservative improvement, comparing the supervised worst case (2 h) against the autopilot upper bound (2.5 h). The upper bound,  $-71.4\%$ , corresponds to the largest improvement, comparing the supervised best case (1 h) against the teleoperated case (3.5 h).

|                           | Teleoperated | Autopilot    | Supervised | Change               |
|---------------------------|--------------|--------------|------------|----------------------|
| <b>Gate Crossing</b>      | 120 s        | 120 s        | 63.16 s    | $-47.37\%$           |
| <b>Parking</b>            | 90 s         | 90 s         | 76.74 s    | $-14.73\%$           |
| <b>Total mission time</b> | 3.5 h        | 1.5 to 2.5 h | 1 to 2 h   | $-20\%$ to $-71.4\%$ |

Table 6.5: Procedure-level and mission-level performance comparison across operational modes.

## 6.5 Mission Level Timeline and Qualitative Interpretation

Figure 6.5 illustrates a mission level timeline for two robots (MIRA 1 and MIRA 2) over a 2 hour window. The Gantt style representation shows how the overall mission is structured as long navigation segments punctuated by short local procedures, namely gate crossing, exit from the charging station, and docking in the charging station.

The visualization supports three practical observations. First, navigation dominates the time budget, which indicates that mission duration is primarily driven by global displacement rather than by the duration of individual local maneuvers. Second, gate crossing appears as a brief, recurrent event, consistent with the measured procedure level execution time, and its automation therefore provides cumulative benefits when repeated multiple times. Third, charging related actions are concentrated at the mission boundaries, reflecting the role of the charging station as an initial and terminal condition in the mission sequence. In addition, recovery actions are localized in time and can be interpreted as discrete mission interruptions. When they occur, their impact on mission duration is driven by the recovery strategy selected by the Mission Manager, typically repetition of a local action, teleoperation fallback, or system restart, as summarized in Table 5.1.

Overall, this figure complements Table 6.5 by providing an intuitive system level view of where time is spent during the mission, and how procedure level improvements translate into mission level impact.

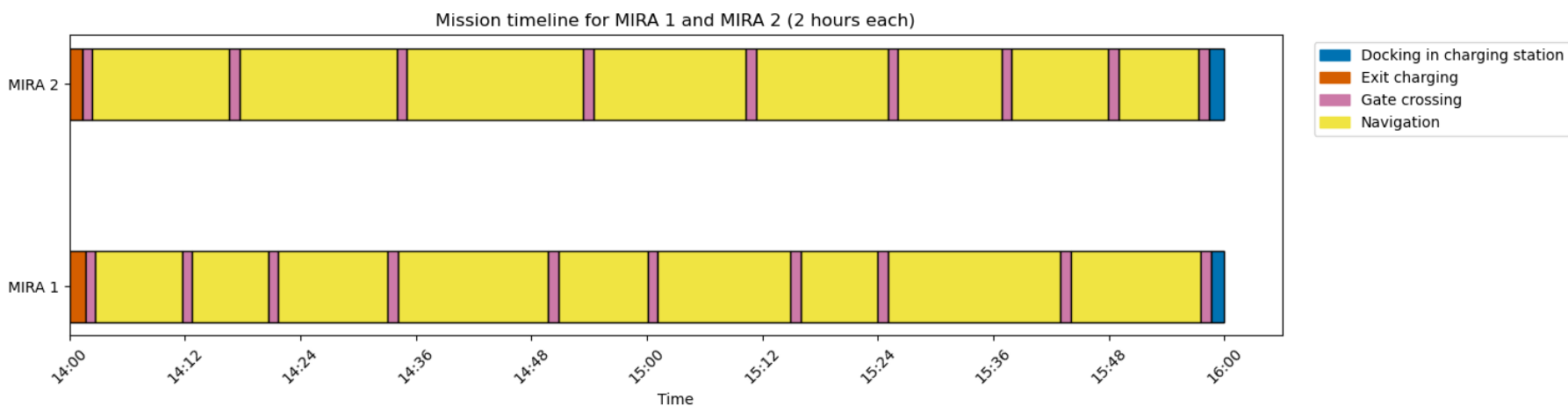


Figure 6.5: Estimated mission timeline for MIRA1 and MIRA2.

## Chapter 7. Conclusions and Future Work

This chapter summarizes the main outcomes of the thesis in the context of supervised mission-level autonomy for SPS radiation surveys with the MIRA robot. It then outlines concrete directions to further increase autonomy, robustness, and operational readiness for long-term deployment in the SPS tunnel.

### 7.1 Conclusions

This thesis presented the design and implementation of mission-level supervised autonomy enabling the MIRA robot to execute full gamma radiation surveys in the CERN SPS tunnel with minimal operator intervention. A key outcome is the development and deployment of the mission logic and autonomous procedures that enabled the first complex robotic radiation survey at CERN to be executed end-to-end under supervision rather than continuous teleoperation. The main contribution is the automation of the survey workflow through a hierarchical Mission Manager that formalizes mission sequencing, procedure invocation, supervision, and recovery. Vision-based servoing is used as an enabling component for specific constrained manoeuvres, while the primary outcome of this work is the operational survey automation achieved by coordinating existing navigation with autonomous local procedures and structured operator supervision.

#### **End-to-end deployment of the first complex autonomous robotic radiation survey at CERN**

This work developed, integrated, and validated the complete mission logic and autonomous procedure set required to run a radiation survey in the SPS tunnel with minimal human intervention. The contribution goes beyond individual modules by delivering an operational end-to-end survey execution, where supervision is limited to safety-relevant validations and recovery decisions instead of continuous teleoperation and manual phase sequencing.

#### **Mission-level automation for SPS surveys with MIRA**

A hierarchical state machine architecture (`RPSurveySPS`) was developed to coordinate the SPS survey as a structured mission rather than a collection of manually sequenced modules. The Mission Manager coordinates long-duration navigation under an existing autopilot baseline with mission-defining constrained events, in particular safety-gate crossings and automatic return-to-charge. It provides deterministic mission progression through explicit states, event-driven transitions, and guard conditions related to the mission context, while keeping the operator in the loop when safety-relevant steps or recovery decisions are required. The system enhances traceability, lowers operator-dependent variability, and enables uniform execution across several surveys by isolating events, guards, actions, and sub-procedures.



### **Autonomous procedures integrated as mission behaviours**

Two environment-specific procedures were implemented and integrated as sub-state machines: **GateCrossing** and **EnteringCharging**. These behaviors are handled as mission components with clear entrance conditions, completion criteria, and failure modes in this work rather than as stand-alone scripts. They are integrated into navigation phases by the Mission Manager, who also ensures their exclusive execution, a smooth transition back to nominal survey operation, and consistent management of unexpected results. This integration is a key factor for operational autonomy because it removes the need for manual phase switching and decision making during long missions with recurrent constrained events.

### **Visual Servoing as a reusable and modular capability within the CRF**

A visual servoing module was designed and implemented from scratch and integrated into the CERN Robotics Framework as a reusable capability. The final component operates only on a position estimate between the camera and the target and is not dependent on any particular visual landmark, in contrast to a process linked to a single visual marker. The module was integrated in the CRF so that it can be reused with different CERN robots and different camera setups, including eye-in-hand and eye-to-hand configurations, by relying on standard CRF transforms between reference frames. Within this thesis, the module was instantiated for marker-based PBVS alignment to provide the precision required in narrow areas that cannot be handled robustly by nominal navigation, in particular alignment with the gate cut-out before traversal and repeatable positioning during docking in the charging station. Experimental results confirm consistent convergence from multiple initial conditions, delivering the residual pose accuracy required before committing to the corresponding constrained motions, while preserving portability to other robots, environments, and visual targets.

### **Structured supervision, error handling, and operational impact**

A central requirement in accelerator environments is deterministic behaviour under faults and traceable operator decisions. This thesis implemented a layered strategy in which anomalies are handled locally within the active procedure when possible and escalated to a global recovery layer otherwise. Recovery actions are standardized (e.g., retrying a manoeuvre, switching to teleoperation, or restarting after safety-critical events), making failures bounded, predictable, and auditable, and enabling systematic testing of consistent failure semantics.

### **Reliability, operator workload reduction, and efficiency**

The supervised autonomy introduced in this work standardizes the execution and recovery of critical manoeuvres and improves operational reliability. Gate crossing achieved a 100% success rate over 50 trials from multiple starting positions, with minor contacts observed only under very poor illumination. Parking and docking achieved a 98% success rate under the same protocol, with the single failure caused by mecanum wheel slip on the station ramp, indicating a limitation driven by traction and ramp interaction rather than Mission Manager or PBVS software correctness.

Operationally, the Mission Manager shifts interaction from continuous teleoperation and manual sequencing to monitoring and discrete supervision decisions. At procedure level, gate crossing

time is reduced from 120 s (teleoperated) to 63.16 s (supervised), a 47.37% reduction, and parking time is reduced from 90 s to 76.74 s, a 14.73% reduction. At mission level, these effects translate into a substantial reduction in total mission duration by removing operator-dependent phase switching and bounding recovery overhead. As summarized in Table 6.5, supervised execution reduces the estimated mission time from 3.5 h (teleoperated) to 1 to 2 h, and remains at least 20% faster than the worst-case autopilot estimate (2.5 h) under the same mission structure.

## Summary

Overall, this thesis demonstrates that mission-level supervised autonomy for SPS radiation surveys can be achieved by combining a hierarchical Mission Manager with dedicated autonomous procedures for safety-gate crossing and automatic return-to-charge. In addition, it delivers a modular visual servoing capability integrated into the CRF, designed to be reusable across different robots, camera configurations, and visual beacons, and instantiated here to enable precise alignment within those procedures. The main contribution remains the automation of the survey mission itself, including sequencing, supervision, and recovery, enabling MIRA to execute long tunnel surveys with reduced operator intervention while preserving safety, traceability, and operational consistency.

## 7.2 Future Work

Several extensions can further increase autonomy, robustness, and operational readiness, and facilitate transfer of the proposed approach to other CERN facilities and robotic platforms.

### Extension of mission automation to other facilities and platforms

The Mission Manager logic can be reused to automate inspection and survey missions that are currently teleoperated in other CERN infrastructures. A concrete next step is the transfer to LHC operations using the Train Inspection Monorail (TIM) and the TIM Arm, where mission-level sequencing, supervision, and recovery could coordinate monorail motion with arm-based procedures for recurring tasks such as radiation surveys and autonomous detection and localization of Beam Loss Monitors (BLMs). Figure 7.1 illustrates the TIM platform envisioned as a target for this transfer.



Figure 7.1: Train Inspection Monorail (TIM) platform for LHC tunnel operations, extensible with additional payloads such as a robotic arm (TIM Arm) [34].

### Full integration with the operator GUI

While the current system supports supervised execution through explicit operator validation and recovery logic, future work should integrate the Mission Manager directly into the existing operator GUI used for MIRA gamma radiation surveys. This includes mission state and procedure progress visualization, live perception feedback relevant to local procedures, standardized recovery prompts, and structured logging and replay tools for post-mission analysis. Figure 7.2 shows the current teleoperation-oriented interface, which can be extended to expose mission-level status and supervision actions.



Figure 7.2: Current operator GUI used for teleoperated gamma radiation surveys with MIRA.

### Robustness under degraded conditions

The experiments highlighted two dominant sources of degradation: poor illumination affecting visual alignment and wheel slip on ramps during docking. Robustness can be improved by combining perception mitigation (e.g., active lighting and camera parameter management) with slip-aware motion strategies and traction monitoring during ramp traversal. In parallel, mechanical adaptations of wheel or ramp surfaces should be considered, since the dominant failure mode observed is driven by traction and contact interaction.

### Autonomous manipulation procedures enabled by visual servoing

A natural next step is to expand the procedure library beyond navigation and docking to arm-based maintenance tasks, such as connector handling, valve actuation, or screwing and simple assembly operations. The visual servoing module developed in this thesis provides a reusable alignment capability that can be instantiated for such tasks with different visual targets and camera configurations, and complemented with force or torque feedback when contact is required. Figure 7.3 shows an example of visual-servoing-based screwing, representative of the type of manipulation procedures that could be integrated under the same supervision and recovery structure.

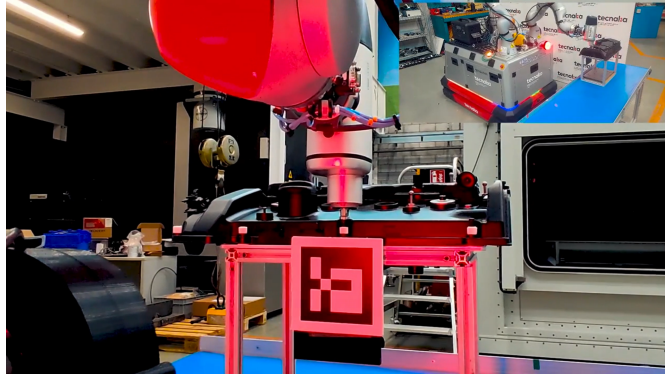


Figure 7.3: Example of visual-servoing-based screwing from the literature [24].

## REFERENCES

- [1] CERN. *The Super Proton Synchrotron (SPS)*. Accessed: 2025-07-07. 2024. URL: <https://home.cern/science/accelerators/super-proton-synchrotron>.
- [2] Ana Lopes. *SPS experiments are back in action. Experiments restart at the Super Proton Synchrotron, CERN's second-largest accelerator*. CERN. Aug. 2021. URL: <https://home.cern/news/news/experiments/sps-experiments-are-back-action> (visited on 12/02/2025).
- [3] CERN. *Alara rule applied to interventions at CERN*. Accessed: 2025-07-07. 2017. URL: <https://cern.ch/alara>.
- [4] David Forkel et al. "Telerobotic Radiation Protection Tasks in the Super Proton Synchrotron using Mobile Robots". In: Jan. 2022, pp. 451–458. DOI: 10.5220/0011276600003271.
- [5] European Organization for Nuclear Research (CERN). *The CERN accelerator complex, layout in 2022*. <https://cds.cern.ch/images/CERN-GRAPHICS-2022-001-1>. Image by Ewa Lopienska, taken and uploaded 3 February 2022. Licencia: ©CERN, uso no comercial. Feb. 2022.
- [6] David Forkel et al. "Mobile Robots for Teleoperated Radiation Protection Tasks in the Super Proton Synchrotron". In: *Informatics in Control, Automation and Robotics*. Ed. by Giuseppina Gini et al. Cham: Springer International Publishing, 2023, pp. 65–82. ISBN: 978-3-031-48303-5.
- [7] David Forkel et al. "Towards Autonomous Execution of Safe and Precise Robotic Radiation Surveys in the Super Proton Synchrotron". In: *Journal of Intelligent Robotic Systems* (Dec. 2025). DOI: 10.1007/s10846-025-02342-0.
- [8] Haotian Zhou et al. "A Robot Architecture of Hierarchical Finite State Machine for Autonomous Mobile Manipulator". In: Aug. 2017, pp. 425–436. ISBN: 978-3-319-65297-9. DOI: 10.1007/978-3-319-65298-6\_39.
- [9] Markus Bajones et al. "Hobbit: Providing Fall Detection and Prevention for the Elderly in the Real World". In: *Journal of Robotics* 2018.1 (2018), p. 1754657. DOI: <https://doi.org/10.1155/2018/1754657>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1155/2018/1754657>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1155/2018/1754657>.
- [10] Paloma De La Puente et al. "Grasping Objects From the Floor in Assistive Robotics: Real World Implications and Lessons Learned". In: *IEEE Access* 7 (2019), pp. 123725–123735. DOI: 10.1109/ACCESS.2019.2938366.
- [11] Markus Bajones et al. "Results of Field Trials with a Mobile Service Robot for Older Adults in 16 Private Households". In: *J. Hum.-Robot Interact.* 9.2 (Dec. 2019). DOI: 10.1145/3368554. URL: <https://doi.org/10.1145/3368554>.
- [12] Guillermo GP-Lenza et al. "A Methodology for Designing Knowledge-Driven Missions for Robots". In: *2024 7th Iberian Robotics Conference (ROBOT)*. 2024, pp. 1–6. DOI: 10.1109/ROBOT61475.2024.10796896.

- [13] J.D. Carbou, D. Andreu, and P. Fraisse. “Events as a key of an autonomous robot controller”. In: vol. 35. 1. 15th IFAC World Congress. 2002, pp. 223–228. DOI: <https://doi.org/10.3182/20020721-6-ES-1901.00849>. URL: <https://www.sciencedirect.com/science/article/pii/S1474667015392703>.
- [14] Yisoo Lee et al. “Toward Reactive Walking: Control of Biped Robots Exploiting an Event-Based FSM”. In: *IEEE Transactions on Robotics* 38.2 (2022), pp. 683–698. DOI: 10.1109/TR0.2021.3088062.
- [15] Stefan Leutenegger et al. “Keyframe-Based Visual-Inertial Odometry Using Nonlinear Optimization”. In: *The International Journal of Robotics Research* 34 (Feb. 2014). DOI: 10.1177/0278364914554813.
- [16] Alejandro Marzinotto et al. “Towards a unified behavior trees framework for robot control”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 5420–5427. DOI: 10.1109/ICRA.2014.6907656.
- [17] Ali Agha et al. “NeBula: TEAM CoSTAR’s Robotic Autonomy Solution that Won Phase II of DARPA Subterranean Challenge”. In: *Field Robotics* 2 (2022), pp. 1432–1506. DOI: 10.55417/fr.2022047.
- [18] S. Hutchinson, G.D. Hager, and P.I. Corke. “A tutorial on visual servo control”. In: *IEEE Transactions on Robotics and Automation* 12.5 (1996), pp. 651–670. DOI: 10.1109/70.538972.
- [19] François Chaumette, Seth Hutchinson, and Peter Corke. “Visual Servoing”. In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. Cham: Springer International Publishing, 2016, pp. 841–866. ISBN: 978-3-319-32552-1. DOI: 10.1007/978-3-319-32552-1\_34. URL: [https://doi.org/10.1007/978-3-319-32552-1\\_34](https://doi.org/10.1007/978-3-319-32552-1_34).
- [20] Sergio Garrido-Jurado et al. “Automatic generation and detection of highly reliable fiducial markers under occlusion”. In: *Pattern Recognition* 47 (June 2014), pp. 2280–2292. DOI: 10.1016/j.patcog.2014.01.005.
- [21] OpenCV Development Team. *Detection of ArUco Markers*. [https://docs.opencv.org/4.x/d5/dae/tutorial\\_aruco\\_detection.html](https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html). Accessed: 2025-10-31. 2025.
- [22] Jesse S. Wynn and Timothy W. McLain. “Visual Servoing with Feed-Forward for Precision Shipboard Landing of an Autonomous Multirotor”. In: *2019 American Control Conference (ACC)*. 2019, pp. 3928–3935. DOI: 10.23919/ACC.2019.8814694.
- [23] Victor Fernandez-Ayala et al. *Robust Visual Servoing under Human Supervision for Assembly Tasks*. Apr. 2025. DOI: 10.48550/arXiv.2504.12506.
- [24] Javier Gonzalez Huarte and Aitor Ibarguren. “Visual Servoing Architecture of Mobile Manipulators for Precise Industrial Operations on Moving Objects”. In: *Robotics* 13 (May 2024), p. 71. DOI: 10.3390/robotics13050071.
- [25] Mario Di Castro, Manuel Ferre, and Alessandro Masi. “CERNTAURO: A Modular Architecture for Robotic Inspection and Telemanipulation in Harsh and Semi-Structured

- Environments”. In: *IEEE Access* 6 (2018), pp. 37506–37522. DOI: 10.1109/ACCESS.2018.2849572.
- [26] Jorge Playán Garai. “System for the high-level mission control of robotic tasks”. Developed at CERN, Mission Manager module. MA thesis. Universidad Politécnica de Madrid, 2021.
  - [27] M. Di Castro et al. “i-TIM: A Robotic System for Safety, Measurements, Inspection and Maintenance in Harsh Environments”. In: *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. 2018, pp. 1–6. DOI: 10.1109/SSRR.2018.8468661.
  - [28] Kris Jusiak. *Boost.Ext SML Tutorial*. 2025. URL: <https://boost-ext.github.io/sml/tutorial.html> (visited on 07/21/2025).
  - [29] *Boost::ext.SML Documentation: UML vs SML*. [https://boost-ext.github.io/sml/uml\\_vs\\_sml.html](https://boost-ext.github.io/sml/uml_vs_sml.html). Accessed: 2026-01-04.
  - [30] Markus Herb. *Kalman Filter Library (header-only)*. <https://github.com/mherb/kalman>. Technical University of Munich. 2015.
  - [31] Ho Chuen Kam, Ying Kin Yu, and Kin Hong Wong. “An Improvement on ArUco Marker for Pose Tracking Using Kalman Filter”. In: *2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. 2018, pp. 65–69. DOI: 10.1109/SNPD.2018.8441049.
  - [32] E. Mouragnon et al. “Real Time Localization and 3D Reconstruction”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. Vol. 1. 2006, pp. 363–370. DOI: 10.1109/CVPR.2006.236.
  - [33] Sang Yoon, Seong-Bae Park, and Jong Kim. “Kalman Filter Sensor Fusion for Mecanum Wheeled Automated Guided Vehicle Localization”. In: *Journal of Sensors* 2015 (July 2015), pp. 1–7. DOI: 10.1155/2015/347379.
  - [34] Pelle De Brabander. *TIM robot in the LHC tunnel*. CERN Document Server, record CERN-PHOTO-202011-145-10. Nov. 2020. URL: <https://cds.cern.ch/images/CERN-PHOTO-202011-145-10> (visited on 01/26/2026).

## ANNEXES

### Annex I - Sustainable Development Goals (SDGs)

This thesis contributes to the long-term development of supervised autonomous robotic operations in CERN facilities. While the work is primarily technical and focuses on mission-level automation, structured supervision, and recovery, it aligns with broader Sustainable Development Goals (SDGs) by improving the safety, reliability, and efficiency of radiation survey operations in a hazardous underground environment.



Figure 7.4: Sustainable Development Goals (SDGs).

The most relevant SDGs for this work are:

- **SDG 3: Good Health and Well-Being.** The automation of radiation surveys reduces the need for personnel to enter radiation-exposed and physically constrained tunnel areas, supporting safer working practices consistent with ALARA principles. In this sense, the contribution is indirect but operationally relevant, since robotics enables risk reduction during routine monitoring campaigns.
- **SDG 8: Decent Work and Economic Growth.** The supervised autonomy framework reduces operator workload by shifting from continuous teleoperation to lightweight monitoring and discrete supervision decisions. This improves occupational safety and helps make long missions less dependent on sustained human attention, which is important for operational sustainability in control-room environments.
- **SDG 9: Industry, Innovation and Infrastructure.** The proposed hierarchical Mission Manager and the integration of autonomous procedures constitute an advanced robotics solution for operating and maintaining critical scientific infrastructure. By improving repeatability, traceability, and fault handling, the system supports more reliable long-duration operations in complex environments.



- **SDG 12: Responsible Consumption and Production.** By improving mission efficiency and standardizing recovery actions, the system reduces time losses and unnecessary repetitions during survey execution. This contributes to more efficient use of operational resources, including battery cycles and tunnel occupation time.

## **Annex II - Impact Evaluation and Ethical-Legal Considerations**

This annex summarizes the main environmental, social, and ethical-legal considerations associated with deploying supervised autonomous robotic surveys in the SPS tunnel at CERN. The evaluation is framed around the operational goal of reducing personnel exposure, improving repeatability, and ensuring safe behaviour through supervised autonomy.

### **Environmental and energy impact**

The contribution of this work is indirect, but operationally relevant. By standardizing mission sequencing and automating constrained manoeuvres (gate crossing and docking), the system reduces time losses due to repeated teleoperated attempts and inconsistent recovery decisions. In long surveys, these improvements can reduce mission duration and therefore energy consumption per survey, including battery usage and recharge cycles. The largest energy and time savings are expected at mission scale, where repeated gate crossings and recovery events accumulate.

### **Social and operational impact**

The primary societal impact in this context is safety. SPS radiation surveys are performed in a hazardous underground environment, and the mission-level autonomy developed in this thesis contributes to reducing the need for personnel to enter the tunnel during routine operations, consistent with ALARA principles. At the control-room level, the Mission Manager reduces operator workload by shifting the operator role from continuous teleoperation and manual phase switching to lightweight monitoring and discrete supervision decisions. This improves repeatability and reduces stress and fatigue during long missions, while preserving accountability through supervised execution.

### **Economic and organizational impact**

Improved repeatability and reduced manual intervention can increase survey throughput and reduce the operational effort required to execute routine monitoring campaigns. In addition, the modular architecture supports incremental extension of the procedure library and mission logic without requiring a complete redesign of the system, which lowers the cost of future developments and facilitates maintenance.

### **Legal and compliance considerations**

A real deployment must comply with CERN safety rules and internal operational procedures for robotic systems in restricted areas. Supervised autonomy is aligned with these constraints because safety-relevant transitions and recovery actions can be explicitly authorized by an operator, and mission execution remains traceable through logs and state transitions. The use of visual sensors requires appropriate handling of recorded data. In this work, the cameras are used primarily for infrastructure-referenced perception, specifically *ArUco*-based alignment at safety gates and charging stations, rather than for observing people. In the SPS context, privacy risks are limited compared to public environments, but data management should still follow CERN policies regarding retention, access control, and secure storage. Finally, third-party software and

open-source components must be used in compliance with their licenses, and proprietary hardware and interfaces must be integrated respecting usage and intellectual property agreements.

### **Ethical considerations**

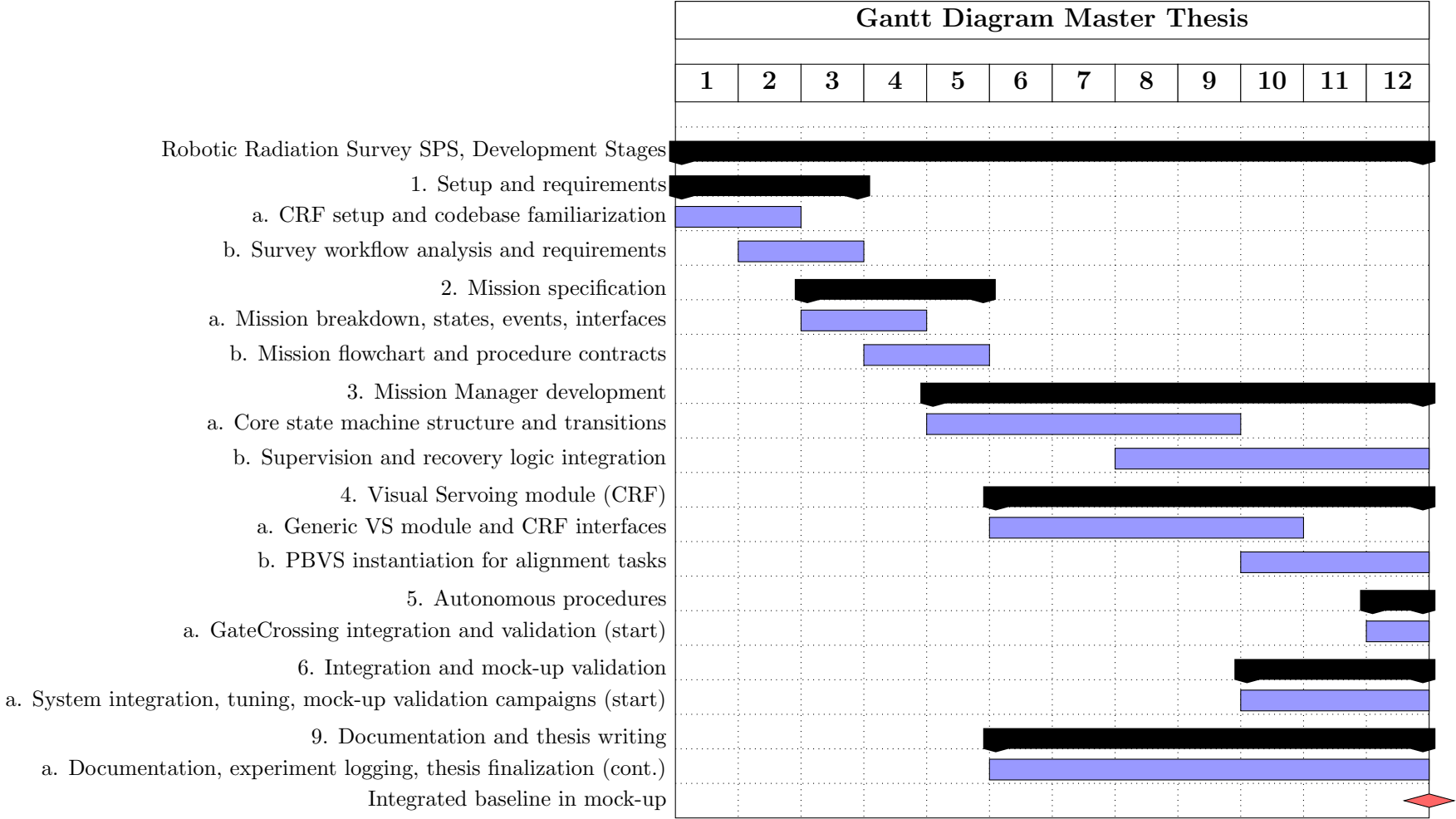
The system is designed around three principles that are directly reflected in the Mission Manager architecture. First, *transparency* is supported through explicit FSM states, logged events, and reproducible recovery decisions. Second, *accountability* is preserved by supervised execution, where the operator retains authority over critical transitions and recovery when required. Third, *safety* is prioritized through deterministic failure handling, safe fallback behaviours, and conservative escalation of faults when local recovery is not possible.

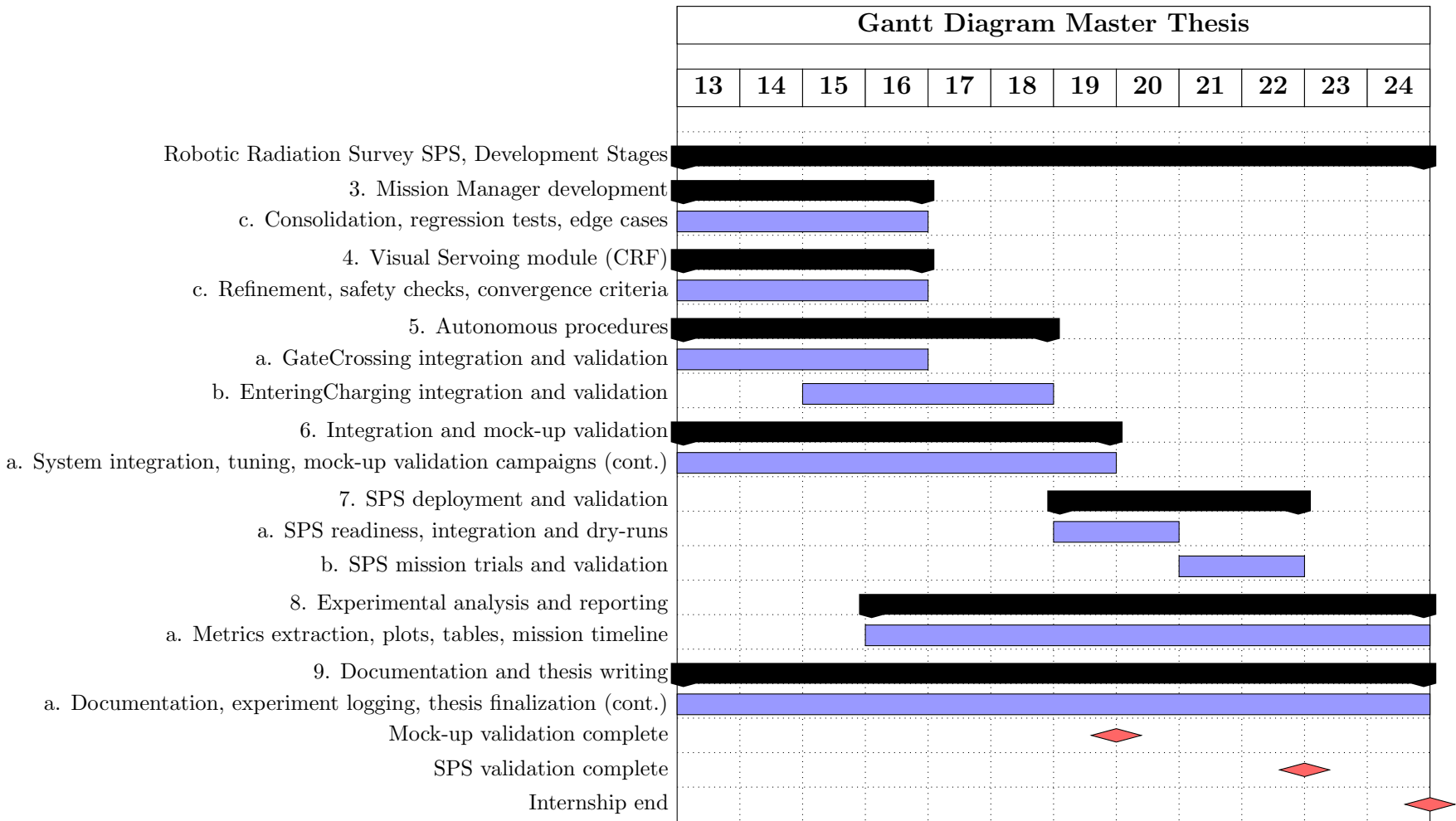
## **Annex III - Temporal Planning and Execution Budget**

### **Temporal Planning**

The following Gantt charts summarize the project plan over a 24-week internship period, from 01-Jul-2025 to 19-Dec-2025. The overall effort was bounded to approximately 380 person-hours, corresponding to about 48 working days at 8 hours per day, distributed over the internship period depending on access windows, integration availability, and validation campaigns.

*The full Gantt chart is shown on the following pages in landscape format.*





**Execution Budget**

| <b>EXECUTION BUDGET</b>                 |                                                                                              |            |                        |                       |                    |
|-----------------------------------------|----------------------------------------------------------------------------------------------|------------|------------------------|-----------------------|--------------------|
| <b>1. Material</b>                      |                                                                                              |            |                        |                       |                    |
| <b>1.1 Software</b>                     |                                                                                              |            |                        |                       |                    |
| <b>Code</b>                             | <b>Description</b>                                                                           | <b>Qty</b> | <b>Unit cost (EUR)</b> | <b>Subtotal (EUR)</b> | <b>Total (EUR)</b> |
| 1.1.1                                   | CRF modules, C++ toolchain, Boost, build system (in-house / open-source)                     | 1          | 0.00                   | 0.00                  |                    |
| 1.1.2                                   | Computer vision libraries and utilities (OpenCV, ArUco, logging, plotting)                   | 1          | 0.00                   | 0.00                  |                    |
| 1.1.3                                   | Documentation and reporting tools (LaTeX, Python environment)                                | 1          | 0.00                   | 0.00                  |                    |
| <b>Software total</b>                   |                                                                                              |            |                        |                       | <b>0.00</b>        |
| <b>1.2 Hardware</b>                     |                                                                                              |            |                        |                       |                    |
| <b>Code</b>                             | <b>Description</b>                                                                           | <b>Qty</b> | <b>Unit cost (EUR)</b> | <b>Subtotal (EUR)</b> | <b>Total (EUR)</b> |
| 1.2.1                                   | MIRA robotic platform (amortized allocation, shared infrastructure)                          | 3          | 40000.00               | 120000.00             |                    |
| 1.2.2                                   | Charging station infrastructure (amortized allocation, shared)                               | 2          | 5000.00                | 10000.00              |                    |
| 1.2.3                                   | SPS gate infrastructure support for surveys (amortized allocation, shared)                   | 1          | 500.00                 | 500.00                |                    |
| 1.2.4                                   | On-board mission computer (LattePanda Sigma SFF)                                             | 1          | 800.00                 | 800.00                |                    |
| 1.2.5                                   | Fiducial markers and mounting accessories (consumables)                                      | 1          | 30.00                  | 30.00                 |                    |
| <b>Hardware total</b>                   |                                                                                              |            |                        |                       | <b>131330.00</b>   |
| <b>2. Labour</b>                        |                                                                                              |            |                        |                       |                    |
| <b>Code</b>                             | <b>Description</b>                                                                           | <b>Qty</b> | <b>Unit cost (EUR)</b> | <b>Subtotal (EUR)</b> | <b>Total (EUR)</b> |
| 2.1                                     | Student work: design, implementation, testing, and documentation                             | 380        | 12.00                  | 4560.00               |                    |
| 2.2                                     | Senior research staff: student supervision, CRF software integration and maintenance support | 160        | 50.00                  | 8000.00               |                    |
| <b>Labour total</b>                     |                                                                                              |            |                        |                       | <b>12560.00</b>    |
| <b>EXECUTION BUDGET (Without Taxes)</b> |                                                                                              |            |                        |                       | <b>143890.00</b>   |

Table 7.1: Execution budget breakdown.

| TOTAL EXECUTION BUDGET                  |  |                       |
|-----------------------------------------|--|-----------------------|
| Description                             |  | Total amount<br>(EUR) |
| Software                                |  | 0.00                  |
| Hardware                                |  | 131,330.00            |
| Labour                                  |  | 12,560.00             |
| Direct Execution Budget (Without Taxes) |  | 143,890.00            |
| Indirect Cost (15%)                     |  | 21,583.50             |
| Subtotal                                |  | 165,473.50            |
| Tax (IVA 21%)                           |  | 34,749.44             |
| <b>TOTAL</b>                            |  | <b>200,222.94</b>     |

Table 7.2: Total execution budget summary.



### **Annex IV - Video Demonstration**

The following video demonstrates the autonomous gate-crossing and parking procedures in a tunnel mock-up under Mission Manager supervision, and concludes with a real operation in the Super Proton Synchrotron:

<https://youtu.be/wVKmsSj0qlk>