



UNIVERSITÀ DEGLI STUDI DI PAVIA

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRICAL, COMPUTER AND BIOMEDICAL ENGINEERING

MASTER'S DEGREE IN INDUSTRIAL AUTOMATION ENGINEERING

MASTER'S THESIS

**Design and Implementation of 3D Reactive
Collision Avoidance Controller for a 7-DOF
Manipulator via Control Barrier Functions**

Candidate:

Sergi Goldsmits Ybarra

Matricola/ID:

554495

Supervisor:

Prof. Antonella Ferrara.

Co - Supervisor:

Nikolas Sacchi, Ph.D.

A.Y. 2025/26

Abstract

While standard position control of robot manipulators is effective for most isolated tasks in the industry, for collaborative robots or robots that share the workspace with other objects this approach may pose a critical safety hazard while performing them.

This thesis aims to approach this problem by designing a velocity actuated position controller that avoids collision with an obstacle of known dimensions and position.

First, a geometric approach to map the obstacles in the joint space is used, creating thus a grouping of joint configurations where the robot cannot go if it intends to avoid collision. Secondly, a reactive logic in the controller that pushes the joints away from the forbidden zone is created, while still completing the task when possible.

Overall, when controlled, the robot geometry will slide along the obstacle area instead of intersecting with it to perform the task. Crucially, this is a hard constraint and if the task proposed cannot be achieved while avoiding the obstacle, it will under no circumstance violate the safety constraints to achieve it.

The controller performance and results have been validated via simulation software (RVIZ) and through live experimental testing using the Franka Emika FR3 robot available in the Intelligent Robotics lab at Università di Pavia. Experimental validation on the 7-DoF Franka Emika FR3 demonstrates hard safety compliance ($h(t) \geq 0$) at a 100 Hz control frequency, with a high end-effector tracking accuracy, +61.7% task duration penalty, and +17.8% energy consumption overhead while performing an imposed experimental task with the safety barrier active.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. Antonella Ferrara, for her guidance and support throughout this project.

I am equally grateful to my co-supervisor, Dr. Nikolas Sacchi, for his practical advice, patience, and technical assistance during the experimental work in the Intelligent Robotics Lab.

My thanks also go to the faculty and staff of the Master's degree program at Università di Pavia for their dedication and for providing a strong academic foundation.

Finally, I want to thank my family. Their constant support, encouragement, and patience made this work possible.

Contents

Chapter 1 - Thesis Introduction	1
1.1. Context and Background	1
1.2. Problem Statement and Motivation	2
1.3. State of the Art and Literature Review	3
1.3.1. Traditional Reactive Avoidance: Artificial Potential Fields(APF)	3
1.3.2. Predictive & Optimization-Based Control: Model Predictive Control (MPC)	3
1.3.3. Safety Filtering via Control Barrier Functions (CBFs) . . .	4
1.3.4. The Spurious Equilibrium Problem in CBF-QPs	4
1.3.5. Workspace Protection vs. Joint-Space Mapping	4
1.4. Proposed Solution and Contributions	5
1.5. Thesis Structure	6
Chapter 2 - Theoretical Background	7
2.1. Introduction and Problem Statement	8
2.2. Kinematic Guidance & Directional Guidance	9
2.3. Dynamic Convergence & Numerical Integration	11
2.4. Redundancy Resolution & Null-Space Control	12
2.5. Control Barrier Functions (CBFs) for Hard Safety	13
2.6. Deadlock Avoidance via Tangential Fields	14
2.7. Layered System Architecture	16
Chapter 3 - Mathematical Framework	17
3.1. Obstacle Modelling in 2D	17
3.1.1. Mathematical Formulation of the Collision Zone in Joint Space	18
3.2. Formulation $h(t)$ of $\mathbf{h}$ for the CBF	21
3.2.1. Reactive Steering Law	21
3.3. Numerical corrections and Implementation	22
3.3.1. Aggregation of the Links: Soft Minimum	22
3.3.2. Gradient of the Aggregated Barrier	23
3.3.3. Gradient Clipping	23

3.4. Spurious Equilibria in CBF Safety Filters and a Decoupled Resolution	24
3.4.1. Safety-Filter Preliminaries	24
3.4.2. The Spurious-Equilibrium (Deadlock) Problem	25
3.4.3. A Decoupled Escape in the Nominal Field	26
3.4.4. Safety Guarantee and Its Decoupling from the Escape	26
3.5. Collision Avoidance Controller: Extension to the 3D Case and Circulation-Based Deadlock Avoidance	27
3.5.1. Per-Link Barrier in 3D	28
3.5.2. Whole-Arm Barrier Aggregation	30
3.5.3. Barrier Gradient in 3D	30
3.5.4. The Safety Filter in 3D	31
3.5.5. Redundancy Solution	33
3.5.6. Circulation-Based Deadlock Avoidance	34
Chapter 4 - Controller Architecture and Structure	38
4.1. Control Pipeline overview	39
4.1.1. What is ROS2?	40
4.1.2. What is ROS2control?	42
4.1.3. What is libfranka?	43
4.2. Controller pipeline components summary	44
4.3. Controller architecture implementation in ROS2	45
Chapter 5 - Controller safety layer design	46
5.1. Kinematic Model & Robot Representation	48
5.1.1. Model Parsing & Frame Assignment	49
5.1.2. Forward Kinematics & Task Jacobian Extraction	49
5.2. 3D Capisani Angular Barrier & Softmin Aggregation	50
5.2.1. Spatial Link Geometry	50
5.2.2. Softmin Aggregation & Gradient Estimation	51
5.3. Goal-Attenuated Circulation Vector Field	51
5.3.1. Swirl Vector Generation	52
5.3.2. Barrier Proximity & Goal Attenuation Gating	52
5.4. Resolved-Rate Kinematics & Null-Space Projection	53
5.4.1. Damped Task Inversion & Pure Null-Space Projection	53
5.4.2. Nominal Velocity Composition	53
5.5. Unified Kinematic Bounding & Dual QP Safety Filter	54
5.5.1. Dynamic Actuator Bounds	54
5.5.2. Closed-Form Scalar Dual QP	54
5.6. Offline Diagnostic & Validation Pipeline (plot_results.py)	55
5.6.1. Validation & Diagnostic Figures	55
5.6.2. Terminal Verdict Logging	55
Chapter 6 - Results and Discussion	57
6.1. Methodology	58
6.1.1. 2D experiment setup:	59
6.1.2. 3D experiment setup:	63

6.2. Results:	67
6.2.1. 2D experiment results	68
6.2.2. 3D experiment results	75
6.3. Engineering findings and hardware limitations:	86
6.4. Failure modes and research hurdles log:	87
6.5. Results discussion and conclusions:	88
Chapter 7 – Conclusions and Future Work	90
7.1. Summary of contributions	90
7.2. Key Findings and System performance	91
7.3. Limitations and critical assessment	92
7.4. Future Work and Research Directions	93
7.5. Final Words	93
Bibliography	94

List of Figures

1	Image of collaborative workspace. [1]	2
2	Role of the CBF-based filter within the feedback of control scheme for safe robotics systems. [1]	4
3	A simulation of the 6-DoF manipulator avoiding an obstacle. . .	9
4	Description of a generic circular obstacle and a generic manipulator link in the workspace [11]	18
5	Different cases depending on the link length, position and obstacle size and position. [11]	18
6	2D representation of the robot arm and obstacle	20
7	3D representation of the allowed and forbidden configuration representing (q2, q4, q6) in 2D space. These are joints 1,3,5 when using robotics notation.	20
8	CBF barrier representation depending on joint configurations of the 2D case.	20
9	Representation of the circulation mechanism around the obstacle.	36
10	Full controller architecture scheme separating slow and fast update rates and each of the scripts in order of information flow.	39
11	ROS2 control structural scheme in order of information flow. . .	42
12	Physical FR3 at the intelligent robotics lab.	48
13	Real-time representation in RVIZ of the FR3	48
14	RVIZ representation of the FR3 and obstacle in the testing position.	59
15	Initial position of the FR3 during the 2D case lab test.	61
16	RVIZ representation of the FR3 in the 3D case in the starting position.. . . .	64
17	Initial position of the FR3 during the 3D case lab tests.	65
18	Trajectory of the EE during the task where it avoids the obstacle area.	69
19	Evolution of the CBF barrier $h(t)$ during the task, where $h(t)$ stays above 0, signalling safety barrier compliance	69

20	Correction applied to the nominal velocity in rad/s.	69
21	Engagement of the escape latch during the task. If 0 – not active; if 1 – active.	70
22	Trajectory of the EE during the task where it avoids the forbidden area during the task.	71
23	Evolution of the CBF barrier $h(t)$ during the task where $h(t)$ stays above 0.	71
24	Correction applied to the nominal velocity in rad/s.	71
25	Engagement of the escape latch during the task. If 0 – not active; if 1 – active.. . . .	72
26	Trajectory of the EE during the task where it avoids the forbidden area during the task.	73
27	Evolution of the CBF barrier $h(t)$ during the task where it crosses below 0.	73
28	Null correction applied to the nominal velocity in rad/s.	73
29	Trajectory of the EE in 3D during the task where it avoids the forbidden area during the task.	76
30	Evolution of the CBF barrier $h(t)$ during the task where it does not cross below 0.	76
31	Correction applied to the nominal velocity in rad/s and certified constraint.	76
32	Smooth circulation magnitude in 3D.	77
33	Trajectory of the EE in 3D during the ABCD task where it avoids the forbidden area.	78
34	Evolution of the CBF barrier $h(t)$ during the ABCD task where it stays above 0.	78
35	Task completion percentage over time and energy spent during the ABCD task.	79
36	Trajectory of the EE in 3D during the ABCD task where it crosses the forbidden area.	80
37	Evolution of the CBF barrier $h(t)$ during the ABCD where it crosses below 0.	80
38	Task completion percentage over time and energy spent during the ABCD task.	81
39	Trajectory of the EE in 3D during the task where it avoids the forbidden area.	82
40	Evolution of the CBF barrier $h(t)$ during the task where it never crosses below 0.	82
41	Correction applied to the nominal velocity in rad/s.	82
42	Smooth circulation magnitude in 3D.	83
43	Trajectory of the EE in 3D during the task where it crosses the forbidden area.	84
44	Evolution of the CBF barrier $h(t)$ during the task where it crosses below 0.	84
45	Null correction applied to the nominal velocity in rad/s.	84

List of Tables

1	Mathematical Notation and Symbol Definitions	x
2	Scripts and their functions inside the safety layer and controller.	47
3	Code parameters used during the virtual testing in the 2D case using RVIZ.	60
4	Code parameters used during the virtual testing in the 2D case using the real FR3.	62
5	Code parameters used during the 3D RVIZ experiment	64
6	Code parameters used during the real testing in the 3D case using the FR3.	66
7	Comparison of time and energy consumption with and without CBF filtering.	89

Comprehensive Glossary of Robotics, Control & Software Acronyms

APF: Artificial Potential Field

CBF / CBFs: Control Barrier Function / Control Barrier Functions

CLIK: Closed-Loop Inverse Kinematics

Cobot: Collaborative Robot

DLS: Damped Least-Squares

DoF: Degrees of Freedom

EE: End-Effector

IK: Inverse Kinematics

KKT: Karush-Kuhn-Tucker (conditions)

MPC: Model Predictive Control

PI: Proportional-Integral (controller)

PID: Proportional-Integral-Derivative (controller)

QP / QPs: Quadratic Program / Quadratic Programs

SE (3): Special Euclidean Group in 3D Space

SO (3): Special Orthogonal Group in 3D Space

DDS: Data Distribution Service

OSRF: Open Source Robotics Foundation

ROS / ROS 2: Robot Operating System / Robot Operating System 2

RVIZ: ROS2 Visualization

URDF: Unified Robot Description Format

XML: Extensible Markup Language

YAML: YAML Ain't Markup Language

FCI: Franka Control Interface

FR3: Franka Research 3 / Franka Emika FR3 Manipulator

ACC: American Control Conference

CDC: IEEE Conference on Decision and Control

ECC: European Control Conference

ICMA: IEEE International Conference on Mechatronics and Automation

IEEE: Institute of Electrical and Electronics Engineers

IIROS: IEEE/RSJ International Conference on Intelligent Robots and Systems

Table 1: Mathematical Notation and Symbol Definitions

Symbol	Description	Domain / Units
Kinematics & Manipulator States		
n, m	Active degrees of freedom ($n = 7$) and task space dimension ($m = 3$)	Dimensionless
$q, \dot{q}, \ddot{q} \in \mathbb{R}^n$	Joint configuration, velocity, and acceleration vectors	rad, rad/s, rad/s ²
$\dot{q}_{\text{nom}}, \dot{q}_{\text{safe}}, \dot{q}_{\text{pub}}$	Nominal, QP-certified, and low-pass published velocity commands	rad/s
$x, p_{\text{ee}}, e(t)$	Operational pose, end-effector position, and task error ($x_d - x$)	m, rad
$J(q), J^\dagger, J^+$	Task Jacobian, Damped Least-Squares, and Moore-Penrose pseudoinverses	Matrices
$P_N(q) \in \mathbb{R}^{n \times n}$	Orthogonal null-space projection operator matrix ($I_n - J^+ J$)	Matrix $\mathbb{R}^{n \times n}$
Obstacle Geometry & Joint-Space Mapping		
V, R, r_{eff}	Obstacle center, physical radius, and effective radius ($R + \text{buffer}$)	m
P_i, l_i, d_i	Origin position, length of link i , and distance to obstacle center V	m
$Q_i^W, \delta_i, \mathcal{CB}_i$	Critical angle, angular half-width, and 2D forbidden joint interval	rad, Set
$\hat{u}_i, \hat{n}_i, \phi_i$	Link unit vector, line-of-sight unit vector, and 3D geodesic angle	Vector, rad
Control Barrier Functions & Safety Filter		
$\mathcal{S}, \partial\mathcal{S}$	Safe configuration domain (superlevel set) and its boundary set	Sets in $\mathbb{R}^n$
$h_i(q), h(q)$	Link barrier function and aggregated soft-minimum barrier	Dimensionless
$\nabla h(q), g_{\text{max}}$	Joint-space aggregated barrier gradient and norm-clipping limit	rad ⁻¹ , Dim.
$\lambda_{\text{cbf}}, \beta, \mu^*$	Class- $\mathcal{K}$ decay gain, soft-min sharpness, and optimal dual QP scalar	s ⁻¹ , Dim., Dim.
$\mathcal{U}(q, \dot{q}_{k-1})$	State-dependent velocity box admissible under actuator limits	Set in $\mathbb{R}^n$
Deadlock Avoidance & Controller Gains		
v_p, v_{circ}	Proportional task velocity and goal-attenuated circulation velocity	m/s
$t, \hat{k} \in \mathbb{S}^2$	Tangential swirl direction unit vector and circulation rotation axis	Vectors in $\mathbb{R}^3$
$k_c, w_{\text{max}}, h_{\text{act}}, e_0$	Circulation speed scale, max weight cap, activation scale, smoothing radius	Various
$K_p, K_{\text{null}}, \alpha$	Task tracking gain, null-space posture gain, low-pass filter factor	s ⁻¹ , Dim.

Chapter 1 - Thesis

Introduction

1.1. Context and Background

In recent decades, industrial robotics has undergone a major paradigm shift. Traditionally, robotic manipulators operated strictly inside protective safety cages isolated from human operators, executing high-precision repetitive tasks along predefined trajectories using standard position control. While effective for heavy manufacturing, this rigid separation severely limits operational flexibility [1].

Modern industrial automation increasingly relies on collaborative robots (cobots) designed to share their physical workspace with human technicians. By combining the speed, payload capacity, and repeatability of automated manipulators with the cognitive adaptability and problem-solving skills of human operators, collaborative workspaces significantly enhance efficiency in small-scale assembly, logistics, and flexible manufacturing [1].

1.2. Problem Statement and Motivation

Despite its advantages, operating within a shared workspace introduces severe safety risks [1]. Standard feedback control algorithms (e.g., PI or PID architectures) drive a manipulator along a Cartesian trajectory [2], without real-time awareness of structural environment changes. If an unplanned obstacle or human operator enters the path, standard controllers respond to the growing tracking error by delivering increased motor torques. This torque windup creates an increasingly dangerous physical hazard and renders collisions inevitable [1, 3].

In collaborative environments, treating a collision as a simple tracking error is unacceptable, as it directly threatens human safety and structural equipment integrity. Manipulators require active control systems that perceive environmental boundaries and react instantaneously. The primary challenge lies in designing a controller that guarantees collision avoidance as an absolute hard constraint while maximizing task execution quality whenever geometrically possible [1, 4].

Furthermore, incorporating safety constraints impacts core industrial efficiency metrics—specifically task duration and energy consumption. This thesis evaluates the "cost of safety" by comparing task completion times and total energy consumption between unconstrained and safety-constrained manipulator operations [5].



Figure 1: Image of collaborative workspace. [1]

1.3. State of the Art and Literature Review

To unify safety guarantees with real-time task execution, several collision-avoidance strategies have been proposed in robotics literature . This section reviews the historical evolution, advantages, and mathematical limitations of existing reactive and predictive collision-avoidance strategies.

1.3.1. Traditional Reactive Avoidance: Artificial Potential Fields (APF)

Historically, a widely adopted reactive approach is the Artificial Potential Field (APF) method [6] and expanded in trajectory planning literature [7]. APF constructs a synthetic energy landscape across the workspace: target configurations generate attractive potential wells, while physical obstacles exert repulsive potential barriers . The gradient of this scalar field generates virtual force vectors that steer the manipulator away from obstacles in real time [6, 7].

- **Strengths:** APF algorithms are computationally lightweight and easy to execute within low-latency control loops [6, 7].
- **Limitations:** APF methods lack formal safety proofs and suffer notoriously from local minima traps, where opposing attractive and repulsive force vectors cancel out ($\nabla U(q) = 0$), causing the robot to stall before reaching its objective [6]. This issue was directly encountered during the development phase of this work, and a fitting solution is formulated in Chapter 3 and implemented in Chapters 4 and 5.

1.3.2. Predictive & Optimization-Based Control: Model Predictive Control (MPC)

Alternatively, Model Predictive Control (MPC) frames obstacle avoidance as an online constrained optimization problem solved over a discrete, finite time horizon [8]. At each control timestep, MPC predicts future trajectories, evaluates dynamic obstacle constraints, and computes optimal control inputs [8].

- **Strengths:** MPC gracefully incorporates multi-variable state and input bounds, system dynamics, and predictive trajectory paths [8].
- **Limitations:** While effective for complex dynamic trajectories, MPC imposes a heavy computational burden that can struggle to maintain the deterministic high-frequency loop rates (e.g., 1 kHz) required by modern industrial velocity controllers [1, 8].

1.3.3. Safety Filtering via Control Barrier Functions (CBFs)

To combine formal safety guarantees with real-time execution at a fraction of MPC's computational cost, Control Barrier Functions (CBFs) synthesize safety as an instantaneous Quadratic Program (QP) solved at each control cycle. Pioneered by Ames et al. [4] and applied to collaborative robotics by Ferraguti et al. [1], CBFs map safety requirements into forward-invariant safe sets $\mathcal{S}$ [4]. When combined with nominal task control commands, the CBF acts as a real-time safety filter: leaving nominal control inputs untouched when far from obstacles and modifying them only when approaching safe boundary thresholds ($\dot{h}(q) + \gamma(h(q)) \geq 0$) [1, 4].

1.3.4. The Spurious Equilibrium Problem in CBF-QPs

However, standard CBF-QP safety filters can introduce undesirable local equilibria (spurious deadlocks) along safety boundaries under conflicting task demands. Reis et al. [9] proved mathematically that filtering nominal controllers with CBFs through real-time QPs introduces asymptotically stable equilibria along the safety boundary $\partial\mathcal{S}$ [9]. When nominal task forces directly oppose barrier gradients, the filtered control velocity drops to zero, stalling the robot [9]. To prevent local minima deadlocks without violating forward-invariance guarantees, recent formulations inject circulation vector fields orthogonal to barrier gradients ($\nabla h(q)^\top v_{\text{circ}} = 0$) [10].

1.3.5. Workspace Protection vs. Joint-Space Mapping

Additionally, many existing algorithms evaluate collisions exclusively in the 3D Cartesian workspace, requiring continuous inverse kinematics calculations to protect every individual structural link. Mapping obstacles directly into the robot's joint space (Configuration Space) provides a more mathematically elegant formulation, but generating exact analytical joint-space boundaries ($\mathcal{C}_{\text{obs}}$) in real time remains challenging [2, 11].

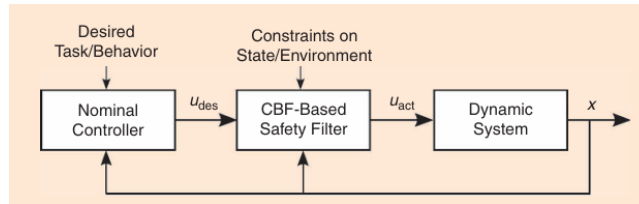


Figure 2: Role of the CBF-based filter within the feedback of control scheme for safe robotics systems. [1]

1.4. Proposed Solution and Contributions

While APFs lack formal safety proofs [6] and MPC requires substantial computational overhead [9], Control Barrier Functions (CBFs) synthesize safety as an instantaneous Quadratic Program (QP) solved at each control cycle [4]. CBFs provide formally provable forward invariance of safe configuration sets at a fraction of MPC's computational cost, making them ideal for high-frequency hard-constraint execution [1, 4].

To overcome traditional safety filter limitations, this thesis presents the design, implementation, and experimental validation of a velocity-actuated position controller integrated with a geometric CBF safety filter [3, 4]. The controller solves a real-time QP to execute joint velocities as close as possible to nominal task commands while strictly enforcing safe boundary conditions [3, 4].

The primary contributions of this work are four-fold:

1. **Geometric Joint-Space Mapping:** Development of an exact analytical approach to map physical workspace obstacles of known dimensions and positions into joint space, constructing continuous forbidden configuration zones [11].
2. **Reactive Control Logic:** Formulation of a velocity-actuated control law using Control Barrier Functions [1, 4] and soft-minimum barrier aggregation [12]. The controller enforces a hard safety boundary that causes the manipulator geometry to slide along obstacle surfaces, prioritizing safety over task completion whenever a conflict occurs [1, 4].
3. **Deadlock Mitigation:** Integration of circulation vector fields orthogonal to barrier gradients to prevent local minima deadlocks without violating forward-invariance guarantees [10, 19].
4. **Experimental Validation and Cost-of-Safety Analysis:** Full system implementation within the `ros2_control` framework [13] using URDF models [14], validated via RViz simulation and physical hardware experiments on a 7-DOF Franka Emika FR3 manipulator [1]. The work provides a quantitative analysis evaluating execution time and energy overhead under safety-constrained conditions [5].

1.5. Thesis Structure

The remainder of this thesis is organized as follows:

- **Chapter 2: Theoretical Background** establishes the mathematical modeling of robotic manipulators [2], forward kinematics, differential Jacobians [2, 3], and other concepts that this thesis builds on.
- **Chapter 3: Mathematical Framework** details the analytical transformation of Cartesian obstacles into forbidden joint-space zones [11], the 3D spatial vector extension [4], and circulation field formulations for deadlock resolution [10].
- **Chapter 4: Controller Architecture and Structure** describes the software pipeline, integrating joint-space obstacle maps and CBF safety logic into the ROS 2 control ecosystem (ROS2_control) [13] and URDF configuration interfaces [14].
- **Chapter 5: Controller safety layer design** outlines the velocity-actuated position control loop [3], redundant null-space task execution [15], damped least-squares inverse kinematics [16], log-sum-exp soft-minimum barrier aggregation [12], and the box-constrained QP filter .
- **Chapter 6: Results and Discussion** presents comparative simulation data in RViz and experimental results from physical testing on the FR3 manipulator [13], evaluating performance against classical proportional velocity controllers in robotics [7] and analysing task completion times and energy consumption [5].
- **Chapter 7: Conclusions and Future Work** summarizes the core theoretical and experimental findings and discusses potential extensions to dynamic, unstructured environments.

This structure represents a gradual path from theoretical framework to synthesis of a real controller tested in real time and real-world conditions. It goes from understanding the robot geometry, kinematics and dynamics, to mapping the workspace in the joint space with the “allowed” and “forbidden” regions to comply with collision avoidance, to designing the controller and testing, to finally discussing the obtained performance and results. The objectives of this thesis will be validated during the results and conclusion chapters, closing the loop in the development process.

Chapter 2 - Theoretical Background

This chapter builds the mathematical and control-theoretic groundwork required to keep a 7-DOF manipulator safe in real time. Rather than treating motion planning and safety as separate steps, the approach here relies on wrapping high-level safety constraints directly around the robot’s kinematic loop. Doing so requires linking differential kinematics with set-invariance theory, ensuring the robot avoids obstacles without compromising its primary trajectory.

The background is structured into the following core sections:

- **Section 2.1: Introduction & Problem Statement** — Shared workspace safety and the shift to streaming joint velocity commands ($\dot{q}_{\text{safe}}$). [1]
- **Section 2.2: Kinematic Guidance & Directional Guidance** — Forward kinematics, Jacobian matrices ($J(q)$), Closed-Loop Inverse Kinematics (CLIK), and pseudoinverse mapping ($J^\dagger(q)$). [2]
- **Section 2.3: Dynamic Convergence & Numerical Integration** — Iterative velocity integration (q_{k+1}) and emergent final configurations (q_∞). [2]
- **Section 2.4: Redundancy Resolution & Null-Space Control** — 4-D null-space projection operators ($P_N(q)$) and secondary ergonomic rest-posture bias ($\dot{q}_0$). [2]
- **Section 2.5: Control Barrier Functions (CBFs) for Hard Safety** — Forward-invariant safe sets ($\mathcal{S}$), linear velocity constraints, and real-time QP filtering under physical actuator limits. [1]

- **Section 2.6: Deadlock Avoidance via Tangential Fields** — Spurious equilibria dynamics, KKT stall conditions, and tangential circulation vector fields (v_{circ}). [7, 9, 10]
- **Section 2.7: Layered System Architecture** — Dual-loop high/low-frequency execution pipelines and hardware saturation consistency.[13]

The chapter defines the precise research gap this thesis fills: combining whole-arm 3D angular CBFs with a smooth deadlock recovery routine, fully integrated into a real-time, actuator-constrained execution stack.

2.1. Introduction and Problem Statement

As introduced in Chapter 1, safe robot manipulation in workspaces shared with human operators presents a fundamental control challenge: the system must reliably track task targets while simultaneously respecting strict safety guarantees against unforeseen collisions [1]. In dynamic environments where humans, obstacles, and secondary robotic agents co-exist, pre-computed offline trajectories quickly become obsolete. To ensure workspace safety without compromising task precision, the control framework must act reactively and modify commands in real time at high execution frequencies [2].

Traditional position-based control architectures rely on sending discrete joint-space position targets (q_{cmd}) to low-level motor drivers [3]. While suitable for isolated industrial cells, this paradigm creates severe limitations in collaborative settings [1]. Position-level inverse kinematics solvers often suffer from multi-solution ambiguity and an inability to gracefully enforce instantaneous state-dependent safety constraints during motion [3]. Moreover, position targets do not naturally support real-time mathematical safety filters, which inherently operate on state rates and velocity bounds [4].

To resolve these limitations, this framework adopts a velocity-level control paradigm [3]. Rather than executing pre-computed joint configurations, the controller continuously streams real-time joint velocity commands ($\dot{q}_{\text{safe}}$) directly to the joint-level driver network. By formulating task tracking as a differential mapping from Cartesian velocity errors to joint velocities, the system can inject active safety constraints directly into the command pipeline [3].

Under this framework, Control Barrier Functions (CBFs) function as minimal-intervention safety filters that operate continuously on the nominal velocity vector $\dot{q}_{\text{nom}}$ [4]. At every control loop tick, an instantaneous Quadratic Program (QP) minimally alters the nominal velocities by solving a constrained optimization problem whenever the robot approaches constraint boundaries [4]. Streaming velocity commands at high frequencies ensures smooth numerical integration, precise target convergence, and deterministic safety enforcement across the manipulator's full kinematic chain [1].

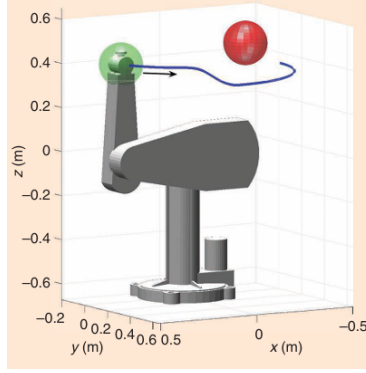


Figure 3: A simulation of the 6-DoF manipulator avoiding an obstacle.

2.2. Kinematic Guidance & Directional Guidance

To establish task-space motion tracking while preserving real-time safety constraints, the control framework utilizes differential kinematics to construct a continuous velocity-level mapping between the manipulator's operational space and joint space [2, 3].

Forward Kinematics and Homogeneous Transformations

The configuration of an n -degree-of-freedom manipulator is mapped from joint space $q \in \mathbb{R}^n$ to operational space via the homogeneous transformation matrix $T_e^b(q) \in SE(3)$ [2, 3]:

$$[T_e^b(q) = \begin{bmatrix} R(q) & p(q) \\ 0_{1 \times 3} & 1 \end{bmatrix} \quad (1)$$

where $p(q) \in \mathbb{R}^3$ denotes the Cartesian end-effector position vector and $R(q) \in SO(3)$ is the rotation matrix [2, 3]. Defining an orientation parameterization vector $\phi(q) \in \mathbb{R}^3$ (e.g., Euler angles), the operational pose is expressed as $x = [p^\top \quad \phi^\top]^\top \in \mathbb{R}^m$ [2, 3].

The instantaneous task-space tracking error $e(t) \in \mathbb{R}^m$ measures the deviation from the desired Cartesian reference trajectory $x_d(t)$ [2, 3]:

$$e(t) = x_d(t) - x(q(t)) \quad (2)$$

Differential Kinematics and the Jacobian Matrix

Differentiating the position and orientation mappings with respect to time yields the differential kinematic relationship:

$$\dot{x} = J(q)\dot{q} \quad (3)$$

where $J(q) \in \mathbb{R}^{m \times n}$ represents the configuration-dependent analytical Jacobian matrix [2, 3]. The Jacobian maps individual joint sensitivity vectors into Cartesian directional velocity vectors, serving as the fundamental transformation for directional guidance [2, 3].

Differential IK (CLIK) vs. Analytical Position IK

Solving inverse kinematics purely at the position level via $q = f^{-1}(x_d)$ presents severe limitations during dynamic collaborative execution [1, 3]:

- **Solution-Branch Discontinuities:** Position-level inverse solvers suffer from multi-solution ambiguity, leading to unpredictable configuration jumps across workspace boundaries [3].
- **Redundancy Resolution:** For redundant manipulators ($n > m$), static position solvers cannot continuously optimize secondary objectives or manage self-motion manifolds on the fly [3].
- **Constraint Enforcement:** Position-level outputs do not naturally accommodate real-time, state-dependent rate and velocity safety bounds [1].

To eliminate these issues, Closed-Loop Inverse Kinematics (CLIK) transforms task-space velocity references frame-by-frame [3]. This continuous differential formulation guarantees smooth joint trajectories, prevents configuration jumps, and produces a velocity vector field suitable for real-time safety filtering [3]

Jacobian Pseudoinverse and Nominal Command Generation

For redundant manipulators ($n > m$), the mapping from desired operational space velocities $v_d \in \mathbb{R}^m$ to nominal joint velocity commands $\dot{q}_{\text{nom}} \in \mathbb{R}^n$ is computed using the Moore–Penrose pseudoinverse $J^\dagger(q) = J^\top (JJ^\top)^{-1}$. Injecting proportional error feedback ensures closed-loop tracking stability[3]:

$$\dot{q}_{\text{nom}} = J^\dagger(q) (\dot{x}_d + K_p e) \quad (4)$$

where $K_p \in \mathbb{R}^{m \times m}$ is a positive-definite gain matrix driving tracking errors exponentially to zero [3]. The output vector $\dot{q}_{\text{nom}}$ provides unconstrained directional guidance, acting as the baseline reference for downstream Control Barrier Function (CBF) safety filtering [1].

2.3. Dynamic Convergence & Numerical Integration

In a velocity-level control architecture, joint positions are not directly commanded; instead, the physical or simulated robot configuration evolves dynamically through continuous numerical integration of the safe joint velocity vector $\dot{q}_{\text{safe}}$ [3].

Iterative Closed-Loop Execution and Discrete-Time Integration

At each discrete control tick k with sampling interval Δt , the controller executes the safe joint velocity vector $\dot{q}_{\text{safe},k}$ generated by the optimization layer in (15). The system updates its joint configuration according to discrete-time numerical integration [3]:

$$[q_{k+1} = q_k + \int_{t_k}^{t_k + \Delta t} \dot{q}_{\text{safe}}(\tau) d\tau \approx q_k + \dot{q}_{\text{safe},k} \Delta t] \quad (5)$$

While the end-effector pose $x(q_k)$ is far from the target pose x_d , the task-space error e_k , defined in (2), drives the nominal velocity command $\dot{q}_{\text{nom},k}$. Closed-loop convergence is achieved asymptotically: as $e_k \rightarrow 0$, the error vanishes ($e \approx 0$), causing the Cartesian velocity demand and resulting joint velocities to decay smoothly to zero ($\dot{q}_{\text{safe}} \rightarrow 0$), bringing the manipulator to rest at the goal [3].

Emergence of Final Joint Configurations (q_∞)

Unlike position-based inverse kinematics solvers that select a static, pre-computed goal posture q_d a priori, the velocity-level framework yields an emergent final joint configuration q_∞ [3]:

$$[q_\infty = q(0) + \int_0^\infty \dot{q}_{\text{safe}}(t) dt] \quad (6)$$

For redundant manipulators ($n > m$), the steady-state configuration q_∞ is path-dependent and dynamically shaped throughout execution by three primary factors [1, 3]:

- **Initial Pose (q_0):** Anchors the trajectory to a specific starting point within the manipulator's continuous self-motion manifold [3].
- **Null-Space Rest-Posture Bias:** Secondary velocity projections $P_N(q)\dot{q}_0$ bias joint configurations toward comfortable ergonomic mid-ranges, joint limit avoidance, or preferred home postures without altering task execution [3].
- **Active Barrier Repulsive Fields:** Real-time modifications imposed by Control Barrier Functions (CBFs) deflect the velocity vector $\dot{q}_{\text{safe}}$ away from workspace obstacles and hazardous state boundaries [4].

Consequently, the final joint configuration q_∞ naturally adapts to transient workspace obstacles encountered during movement, maintaining safety without compromising ultimate Cartesian goal reachability.

2.4. Redundancy Resolution & Null-Space Control

When executing a 3D Cartesian position task ($m = 3$) with a 7-degree-of-freedom manipulator ($n = 7$), the system possesses $n - m = 4$ redundant degrees of freedom [2]. This 4-D null space provides continuous self-motion manifolds, allowing internal joint reconfigurations without affecting the primary end-effector pose [3].

Null-Space Projection Matrix (P_N)

To exploit redundancy without disturbing primary operational tracking, secondary joint-space tasks are filtered through the orthogonal null-space projection operator:

$$P_N(q) = I_{n \times n} - J^\dagger(q)J(q) \quad (7)$$

where $J^\dagger(q)$ is the true Moore–Penrose pseudoinverse [3]. Because $J(q)P_N(q) = 0_{m \times n}$, any arbitrary joint velocity vector $\dot{q}_0$ projected via $P_N(q)$ causes zero instantaneous Cartesian motion at the task level [3].

Secondary Rest-Posture Bias

In this control framework, the secondary velocity vector $\dot{q}_0$ is defined as a proportional joint-space posture controller driving the 7 joints toward an ergonomic reference posture q_{rest} [3]:

$$\dot{q}_0 = K_{\text{null}}(q_{\text{rest}} - q) \quad (8)$$

Combining the primary task tracking command from (4), the null-space projection matrix $P_N(q)$ from (7), and the secondary posture controller from (8) yields the total unconstrained nominal velocity command $\dot{q}_{\text{nom}}$:

$$\dot{q}_{\text{nom}} = J^\dagger(q)(\dot{x}_d + K_p e) + (I - J^\dagger(q)J(q))\dot{q}_0 \quad (9)$$

This decoupled formulation ensures joint limit avoidance and natural posture stabilization during execution without compromising task tracking performance.

2.5. Control Barrier Functions (CBFs) for Hard Safety

To guarantee deterministic collision avoidance while maintaining task execution, the control framework integrates Control Barrier Functions (CBFs) directly into the velocity-level inverse kinematics pipeline.

Safe Configuration Domains

The safe state space of the manipulator is defined as a closed set $\mathcal{S} \subset \mathcal{C}$, constructed as the superlevel set of a continuously differentiable scalar function $h(q) : \mathbb{R}^n \rightarrow \mathbb{R}$:

$$\mathcal{S} = \{q \in \mathbb{R}^n : h(q) \geq 0\} \quad (10)$$

$$\partial\mathcal{S} = \{q \in \mathbb{R}^n : h(q) = 0\} \quad (11)$$

$$\text{Int}(\mathcal{S}) = \{q \in \mathbb{R}^n : h(q) > 0\} \quad (12)$$

In this framework, $h(q)$ models geometric clearance between individual manipulator link segments and workspace obstacles. Points satisfying $h(q) > 0$ correspond to strictly safe configurations, while $h(q) = 0$ represents the physical safety boundary [1, 4, 11].

Forward Invariance & Velocity Linearization

To ensure hard safety, the set $\mathcal{S}$ must be rendered forward invariant: any trajectory starting in $\mathcal{S}$ at $t = 0$ remains within $\mathcal{S}$ for all $t \geq 0$. Applying Nagumo's theorem and extended class $\mathcal{K}$ function conditions, forward invariance is guaranteed if the time derivative satisfies:

$$\dot{h}(q) + \gamma(h(q)) \geq 0, \quad \gamma > 0 \quad (13)$$

Applying the chain rule transforms this condition into a linear inequality in the joint velocity vector $\dot{q}$:

$$\nabla h(q)^\top \dot{q} + \gamma h(q) \geq 0 \quad (14)$$

where $\nabla h(q) = \frac{\partial h(q)}{\partial q} \in \mathbb{R}^n$ is the barrier gradient vector with respect to the active joint configuration. Parameter γ acts as a linear decay rate, dictating how aggressively the controller decelerates the manipulator as it approaches the boundary $h(q) = 0$ [1, 4].

Actuator-Constrained QP Filtering

The nominal command $\dot{q}_{\text{nom}}$ generated by task tracking and null-space redundancy resolution (9) acts as an unconstrained reference. A Quadratic Program (QP) continuously projects $\dot{q}_{\text{nom}}$ onto the admissible safe velocity space while enforcing the forward-invariance CBF condition derived in (14) alongside physical actuator constraints:

$$\dot{q}_{\text{safe}} = \arg \min_{\dot{q} \in \mathbb{R}^n} \frac{1}{2} \|\dot{q} - \dot{q}_{\text{nom}}\|^2 \quad (15)$$

$$\text{subject to } \nabla h(q)^\top \dot{q} \geq -\gamma h(q) \quad (16)$$

$$\dot{q}_{\min}(q, \Delta t) \leq \dot{q} \leq \dot{q}_{\max}(q, \Delta t) \quad (17)$$

The state-dependent box constraints enforce physical actuator limits at sampling interval Δt , accounting simultaneously for joint velocity bounds ($\dot{q}_{\text{limit}}$), acceleration bounds ($\ddot{q}_{\text{limit}}$), and jerk limits ($\dddot{q}_{\text{limit}}$):

$$\dot{q}_{\max} = \min \left(\dot{q}_{\text{limit}}, \dot{q}_k + \ddot{q}_{\text{limit}} \Delta t, \dot{q}_k + \ddot{q}_k \Delta t + \frac{1}{2} \dddot{q}_{\text{limit}} \Delta t^2 \right) \quad (18)$$

$$\dot{q}_{\min} = \max \left(-\dot{q}_{\text{limit}}, \dot{q}_k - \ddot{q}_{\text{limit}} \Delta t, \dot{q}_k - \ddot{q}_k \Delta t - \frac{1}{2} \dddot{q}_{\text{limit}} \Delta t^2 \right) \quad (19)$$

This architecture guarantees that the computed command $\dot{q}_{\text{safe}}$ is minimally perturbed from $\dot{q}_{\text{nom}}$, physically executable by the FR3 drives, and formally certified for continuous safety [1, 4].

2.6. Deadlock Avoidance via Tangential Fields

While Control Barrier Functions (CBFs) strictly guarantee collision avoidance, pure active distance constraints can induce local minima and deadlock conditions in the closed-loop velocity field. To preserve continuous task motion without violating safety boundaries, a tangential circulation field is integrated into the nominal task controller [8,9,10].

Spurious Equilibria & Deadlock Dynamics

In standard CBF-QP safety filters, unwanted steady-state equilibria arise when the nominal task velocity $\dot{q}_{\text{nom}}$ (9) directly opposes the barrier gradient vector $\nabla h(q)$ derived in (14). When an obstacle directly intersects the straight-line path to the goal, the task-space projection of these vectors becomes collinear and opposite:

$$\dot{q}_{\text{nom}} \parallel -\nabla h(q) \implies \dot{q}_{\text{nom}} = -\alpha \nabla h(q), \quad \alpha > 0 \quad (20)$$

As the end-effector nears the safety boundary ($h(q) \rightarrow 0$), the safety constraint in (16) ($\nabla h(q)^\top \dot{q} \geq -\gamma h(q)$) forces the velocity component aligned with $\nabla h(q)$ toward zero. Absent any orthogonal or lateral velocity component in $\dot{q}_{\text{nom}}$, the QP solver in (15) yields $\dot{q}_{\text{safe}} = 0$, trapping the manipulator in an asymptotically stable, unwanted equilibrium short of the goal [9].

Circulation Vector Fields ($\mathbf{v}_{\text{circ}}$)

To break this gradient symmetry, a tangential swirl velocity $v_{\text{circ}} \in \mathbb{R}^3$ is injected into the nominal task space prior to differential kinematic resolution [10]:

$$\dot{x}_{\text{ref}} = \dot{x}_d + K_p e + v_{\text{circ}} \quad (21)$$

The circulation field generates a smooth vortex around the obstacle centre p_{obs} , computed using the cross product of a reference axis n_{ref} and position offset $r = p(q) - p_{\text{obs}}$ [10]:

$$v_{\text{circ}} = \Gamma(h, e) \cdot \left(n_{\text{ref}} \times \frac{r}{\|r\|} \right) \quad (22)$$

To maintain progression toward the target pose, n is aligned with goal tracking vector e , reversing direction if necessary. Magnitude scaling relies on two continuous gating terms [10]:

$$\Gamma(h, e) = v_{\text{max}} \cdot \sigma_{\text{barrier}}(h) \cdot \sigma_{\text{goal}}(e) \quad (23)$$

- **Barrier Proximity Gate (σ_{barrier}):** Confines circulation strictly to the immediate safety boundary ($h(q) \leq h_{\text{thresh}}$), decaying to zero in open regions ($h(q) > h_{\text{thresh}}$) [10].
- **Goal Proximity Fade (σ_{goal}):** Attenuates tangential velocity as the target pose is reached ($e \rightarrow 0$), eliminating orbital limit cycles near the target [10].

Substituting $\dot{x}_{\text{ref}}$ from (21) into the differential IK formulation in (3) generates a deflected reference $\dot{q}_{\text{nom}}$. Because v_{circ} modifies only this unconstrained reference, the downstream CBF-QP safety constraint in (14),(19) remains mathematically unchanged, guaranteeing hard safety while ensuring smooth navigation around obstacles.

2.7. Layered System Architecture

Deploying velocity-level safety filters on physical manipulators requires a multi-rate control hierarchy that bridges high-level geometric safety optimization with high-frequency low-level joint actuation [1, 13].

Real-Time Dual-Loop Execution Pipeline

The system architecture operates via a decoupled multi-rate framework:

- **High-Level Safety & Planning Loop (100 Hz):** Executes within the ROS 2 software stack (ros2_control) [13], handling kinematic updates via Pinocchio, task-space control generation ($\dot{q}_{\text{nom}}$), barrier gradient evaluations ($\nabla h(q)$), and box-constrained QP filtering. It streams certified safe velocity vectors $\dot{q}_{\text{safe}}$ to the hardware interface.
- **Low-Level Actuator Control Loop (1000 Hz):** Runs on the dedicated C++ robot controller (e.g., Franka Control Interface), interpolating commanded velocity setpoints and executing high-frequency closed-loop joint velocity/torque regulation [1].

Actuator Saturation Management & Realization Consistency

Formal CBF safety guarantees rely on the fundamental assumption that the joint velocity realized by the physical actuators matches the safety-certified velocity command [1, 4]:

$$\dot{q}_{\text{realized}}(t) \equiv \dot{q}_{\text{safe}}(t) \quad (24)$$

If low-level hardware safety drivers enforce velocity, acceleration, or jerk clamps that are tighter than the high-level QP box constraints ($\dot{q}_{\text{limit, HW}} < \dot{q}_{\text{max, QP}}$), the physical system realization lags the certified command ($\dot{q}_{\text{realized}} \neq \dot{q}_{\text{safe}}$). This command mismatch causes a breakdown in forward invariance, directly violating the safety condition enforced in (16)

$$[\nabla h(q)]^\top \dot{q}_{\text{realized}} < -\gamma h(q) \quad (25)$$

Under this saturation mismatch, the barrier function can decay below zero ($h(q) < 0$, incurring safety violations) even though the high-level QP controller logs a valid, non-negative constraint ($[\nabla h(q)]^\top \dot{q}_{\text{safe}} \geq -\gamma h(q)$). To maintain strict theoretical consistency, low-level hardware caps must remain strictly looser than the high-level QP box limits, positioning the QP filter as the sole authoritative velocity limiter across the entire control stack [1, 3].

Chapter 3 - Mathematical Framework

To guarantee collision avoidance in both 2D and 3D operational spaces, a rigorous mathematical framework is required to delineate the forbidden configuration space ($\mathcal{C}_{\text{obs}}$) and construct a safe reactive controller. This chapter presents the theoretical foundation and mathematical formulation of the proposed manipulator controller. The approach synthesizes joint-space obstacle mapping techniques from Capisani et al. [11] with Control Barrier Function (CBF) safety filtering derived from Ferraguti et al. [20] and Ames et al. [4, 17].

The formulation begins with the 2D planar case (3-DoF), where workspace circular obstacles map directly to forbidden angular regions in joint space, allowing complete visualization of the configuration space boundaries. Subsequently, the formulation is extended to the 3D spatial domain (7-DoF Franka Emika FR3) [21], generalizing link geometry, barrier aggregation, kinematic redundancy resolution, and deadlock prevention. Prior familiarity with manipulator kinematics, control theory, and CBF-QP filters is assumed.

3.1. Obstacle Modelling in 2D

Following the framework of Capisani et al., a planar manipulator is modelled as a serial chain of n rigid links. For the planar configuration operating in the (X, Z) plane, the Franka Emika FR3 is modeled with $n = 3$ active joints (1, 3 and 5). Each link A_i has length l_i . Workspace obstacles are modeled as circular regions $\mathcal{B} \subset \mathbb{R}^2$ characterized by center position $V \in \mathbb{R}^2$ and radius R (inflated by a safety margin). Mapping $\mathcal{B}$ into joint space yields the forbidden configuration region $\mathcal{CB}_i$, defining joint angles that induce physical contact [11].

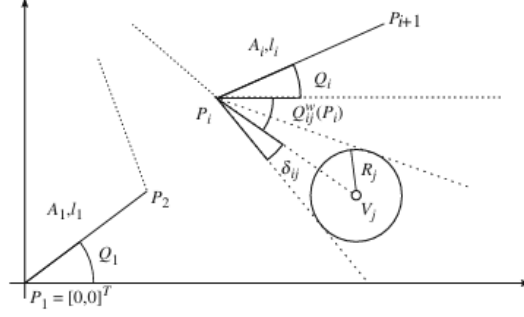


Figure 4: Description of a generic circular obstacle and a generic manipulator link in the workspace [11]

3.1.1. Mathematical Formulation of the Collision Zone in Joint Space

For link A_i , the critical angle Q_i^W defines the joint heading where the link aligns collinearly with vector $\overrightarrow{P_i V}$, pointing directly toward the obstacle center V (where P_i denotes the origin of link A_i). Defining the distance $d_i = \|V - P_i\|$, the angular half-width δ_i of the forbidden cone is defined piecewise [11]:

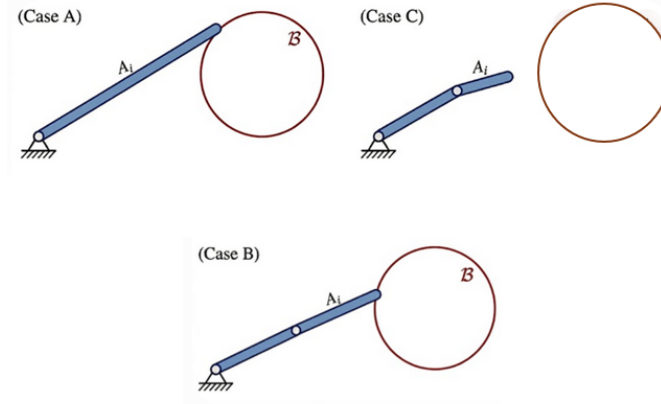


Figure 5: Different cases depending on the link length, position and obstacle size and position. [11]

$$\delta_i = \begin{cases} \arcsin\left(\frac{R}{d_i}\right) & \text{if } l_i^2 \geq d_i^2 - R^2 \quad (\text{Case A: Tangent}) \\ \arccos\left(\frac{l_i^2 + d_i^2 - R^2}{2d_i l_i}\right) & \text{if } (d_i - R)^2 < l_i^2 \leq d_i^2 - R^2 \quad (\text{Case B: Intersection}) \\ 0 & \text{if } l_i^2 \leq (d_i - R)^2 \quad (\text{Case C: Out of reach}) \end{cases} \quad (26)$$

Case A (Tangential Reach): Evaluated via the Pythagorean theorem when the link can extend tangentially to $\mathcal{B}$, yielding $\sin(\delta_i) = R/d_i$.

Case B (Non-Tangential Intersection): Evaluated via the Law of Cosines on the triangle formed by l_i , d_i , and R :

$$R^2 = d_i^2 + l_i^2 - 2d_i l_i \cos(\delta_i) \quad (27)$$

Case C (Unreachable Obstacle): When $l_i \leq d_i - R$, link A_i cannot intersect $\mathcal{B}$ regardless of heading, yielding $\mathcal{CB}_i = \emptyset$ ($\delta_i = 0$).

The forbidden joint interval for link A_i is thus represented as:

$$\mathcal{CB}_i = [Q_i^W - \delta_i, Q_i^W + \delta_i] \quad (28)$$

Where the forbidden configuration region $\mathcal{CB}_i$ is the set of joint angles that result in a collision: $Q_{i,\min} = Q_i^W - \delta_i$, $Q_{i,\max} = Q_i^W + \delta_i$ where the boundary configurations are defined by the critical angle Q_i^W and the angular half-width δ_i .

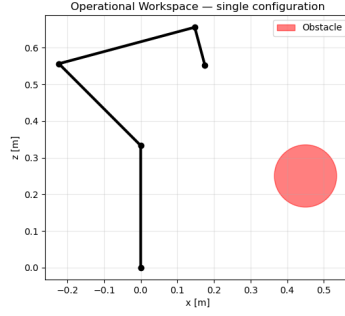


Figure 6: 2D representation of the robot arm and obstacle

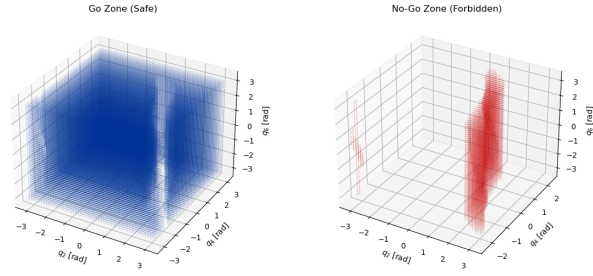


Figure 7: 3D representation of the allowed and forbidden configuration representing (q_2, q_4, q_6) in 2D space. These are joints 1,3,5 when using robotics notation.

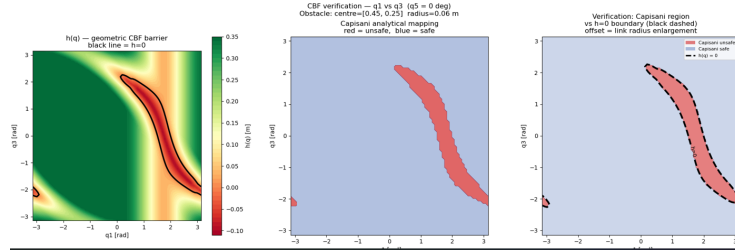


Figure 8: CBF barrier representation depending on joint configurations of the 2D case.

3.2. Formulation $h(t)$ of h for the CBF

To ensure continuous, safe control, a smooth Control Barrier Function $h(Q_i)$ is formulated based on the angular distance from the boundary of $\mathcal{CB}_i$ [20]. To preserve C^1 continuity, the squared angular offset relative to the critical angle Q_i^W and half-width δ_i is defined as:

$$h(Q_i) = (Q_i - Q_i^W)^2 - \delta_i^2 \quad (29)$$

The safe configuration set $\mathcal{S}$ is defined as the superlevel set:

$$\mathcal{S} = \{Q_i \in \mathbb{R} \mid h(Q_i) \geq 0\} \quad (30)$$

Safety ($h(Q_i) \geq 0$) holds whenever $|Q_i - Q_i^W| \geq \delta_i$, placing the link heading strictly outside the forbidden cone $\mathcal{CB}_i$. The boundary $\partial\mathcal{S}$ corresponds to $h(Q_i) = 0$. The barrier gradient with respect to joint heading is :

$$\nabla h(Q_i) = 2(Q_i - Q_i^W) \quad (31)$$

3.2.1. Reactive Steering Law

Forward invariance of $\mathcal{S}$ is maintained using a Control Barrier Function safety filter.

3.2.1.1. General CBF Formulation

Given a nominal velocity command v_p generated by a high-level tracking controller, the safe command v_{cmd} is rendered forward invariant by enforcing $\dot{h} \geq -\alpha(h)$ (where $\alpha(h) = \lambda h$ with $\lambda > 0$). This yields the Quadratic Program (QP) [4, 17]:

$$\min_v \frac{1}{2} \|v - v_p\|^2 \quad \text{s.t.} \quad \nabla h^T v \geq -\lambda h \quad (32)$$

The closed-form analytic projection solution is:

$$v_{\text{cmd}} = v_p - \frac{\min(0, \nabla h^T v_p + \lambda h)}{\|\nabla h\|^2} \nabla h \quad (33)$$

The modification activates only when $\nabla h^T v_p + \lambda h < 0$, deflecting the nominal velocity to remain on or inside $\mathcal{S}$.

3.2.1.2. Specialized Reactive Steering for Robot Manipulators

For this specific planar manipulator, the collision barrier $h(Q_i)$ is defined based on the angular distance to the obstacle cone, defined previously using the Obstacle Modelling paper framework accordingly.

Substituting the per-link gradient $\nabla h(Q_i) = 2(Q_i - Q_i^W)$ into the closed-form projection yields:

$$v_{\text{cmd}} = v_p - \kappa(h) \frac{\min(0, 2\langle Q_i - Q_i^W, v_p \rangle + \lambda h)}{4\|Q_i - Q_i^W\|^2} 2(Q_i - Q_i^W) \quad (34)$$

where $\kappa(h) = 1$ when active and 0 otherwise. This filter acts as a minimally invasive safety layer, preserving v_p during safe operation and deflecting velocity along $\partial\mathcal{S}$ when a collision is imminent.

This controller guarantees that if v_p would push the robot into the forbidden zone **CBi**, the projection operator effectively “deflects” the velocity vector, ensuring that $h(q_i) \geq 0$ is maintained throughout operation under all conditions [20].

3.3. Numerical corrections and Implementation

The previous sections define the mathematical framework of the controller per each link. This section documents how that framework is put into practice and realised in the implementation, without altering the core aspects of it: how the per-link barriers are aggregated into a single value, how its gradient is obtained, and the numerical safeguards applied.

These are implementation choices that recover the framework above within the appropriate limits, they do not change either the safe set or the control law.

3.3.1. Aggregation of the Links: Soft Minimum

Because the arm is safe only when *every* link is safe, whole-arm safety requires evaluating the pointwise minimum over all N link barriers [22]:

$$h_{\text{true}}(q) = \min_{i \in \{1, \dots, N\}} h(Q_i) \quad (35)$$

However, h_{true} is non-differentiable at configurations where the binding (worst-case) link changes. To evaluate a continuous gradient $\nabla h(q)$, the pointwise minimum is approximated using a scaled Log-Sum-Exp (soft-minimum) operator:

$$h(q) = -\frac{1}{\beta} \ln \sum_{i=1}^N \exp(-\beta h(Q_i)), \quad \beta > 0 \quad (36)$$

To avoid arithmetic overflow, $h(q)$ is computed in shifted form using $m = \min_i h(Q_i)$. For any finite scaling parameter $\beta > 0$, $h(q)$ satisfies the two-sided

inequality [12, 18]:

$$\min_i h(Q_i) - \frac{\ln N}{\beta} \leq h(q) \leq \min_i h(Q_i) \quad (37)$$

- **Conservative Safety:** The upper bound $h(q) \leq \min_i h(Q_i)$ guarantees that $h(q) \geq 0 \implies h(Q_i) \geq 0 \forall i$. The soft minimum is strictly conservative by construction.
- **Bounded Underestimation:** For $N = 5$ control points and $\beta = 15$, the conservative offset is bounded by $\frac{\ln 5}{15} \approx 0.107$.

3.3.2. Gradient of the Aggregated Barrier

Applying the chain rule to the soft-minimum aggregation yields a convex combination of per-link gradients [12,23]:

$$\nabla h(q) = \sum_{i=1}^N w_i \nabla h(Q_i), \quad w_i = \frac{\exp(-\beta h(Q_i))}{\sum_{j=1}^N \exp(-\beta h(Q_j))} \quad (38)$$

In practice, $\nabla h(q)$ is evaluated via central finite differences over active joint coordinates:

$$\frac{\partial h}{\partial q_k} \approx \frac{h(q + \epsilon e_k) - h(q - \epsilon e_k)}{2\epsilon} \quad (39)$$

where e_k is the k -th joint unit vector and ϵ is a small step size, reproducing the exact analytic gradient to second order without assembling full link Jacobians explicitly.

3.3.3. Gradient Clipping

Near degenerate kinematic geometries (e.g., when a link is nearly collinear with $\overrightarrow{P_i V}$), $\|\nabla h(q)\|$ can grow excessively large. To prevent numerical instability, the gradient magnitude is bounded while preserving its directional vector:

$$\nabla h \leftarrow \nabla h \cdot \min \left(1, \frac{g_{\max}}{\|\nabla h\|} \right) \quad (40)$$

where $g_{\max}$ is an upper threshold. Bounding magnitude preserves the deflection vector and invariant set $\mathcal{S}$ while preventing control chatter and numerical actuator saturation.

3.4. Spurious Equilibria in CBF Safety Filters and a Decoupled Resolution

This section formalises a well-documented failure mode of control-barrier-functions (CBF) safety filters, the emergence of stable, non-goal equilibria when the target lies on the far side of an obstacle, and presents the resolution adopted in this thesis for the 2D case. First by recalling the safety-filter architecture (Section 3.4.1), characterizing the deadlock geometrically (Section 3.4.2), reviewing the three remedies on which the solution is built and finally describe the decoupled escape (Section 3.4.3) together with forward invariance proof (Section 3.4.4) stating that the controller preserves forward invariance in the safe set $\mathcal{S}$ irrespective of the escape term [9].

3.4.1. Safety-Filter Preliminaries

Consider a kinematic manipulator with configuration $q \in \mathbb{R}^n$ and joint velocity command $\dot{q} \in \mathbb{R}^n$. Operational space task coordinates $x = \text{FK}(q) \in \mathbb{R}^m$ relate to joint rates via the manipulator Jacobian $J(q) = \frac{\partial \text{FK}}{\partial q}$. A proportional goal attraction controller mapped via pseudo-inverse yields:

$$\dot{q}_{\text{nom}} = -K_p J^\dagger \nabla_x V(x), \quad V(x) = \frac{1}{2} \|x_{\text{goal}} - x\|^2 \quad (41)$$

The safety filter enforces forward invariance of $\mathcal{S} = \{q \in \mathbb{R}^n \mid h(q) \geq 0\}$ via the minimum-intervention QP :

$$\dot{q}^* = \arg \min_{\dot{q}} \frac{1}{2} \|\dot{q} - \dot{q}_{\text{nom}}\|^2 \quad \text{s.t.} \quad \nabla h^T \dot{q} \geq -\lambda h \quad (42)$$

yielding the analytical filter :

$$\dot{q}^* = \dot{q}_{\text{nom}} - \frac{\min(0, \nabla h^T \dot{q}_{\text{nom}} + \lambda h)}{\|\nabla h\|^2} \nabla h \quad (43)$$

(43) is the safety-filter form of the nominal tracking controller in (42) when the nominal velocity command $\dot{q}_{\text{nom}}$ already satisfies the safety constraint ($\nabla h^T \dot{q}_{\text{nom}} \geq -\lambda h$), the correction term vanishes and $\dot{q}^* = \dot{q}_{\text{nom}}$; otherwise, the filter projects the command along $\nabla h(q)$ to restore feasibility at the boundary.

3.4.2. The Spurious-Equilibrium (Deadlock) Problem

Combining an attractive goal-seeking controller with a Control Barrier Function safety filter introduces a well-documented structural limitation: the closed-loop system can create unwanted stable equilibrium points along the boundary of the safe set that do not correspond to the target position [4, 9]. These stationary configurations are formally designated as spurious equilibria or deadlocks.

Definition 1 (Spurious Equilibrium): A robot configuration located precisely on the boundary of the safe region is defined as a spurious equilibrium if the filtered joint velocity command drops to zero while the end-effector has not yet reached the intended destination.

Geometric Mechanism of Inverse Collinearity

Deadlocks systematically emerge in scenarios where an obstacle lies directly along the straight-line path between the robot and its target. In these configurations, the goal-attraction field pulls the manipulator directly through the obstacle centre, generating a nominal velocity command that points in exact opposition to the outward safety boundary.

Proposition 1 (Inverse-Collinearity Deadlock): Consider a robot configuration on the safety boundary where the safety barrier function evaluates to zero. If the nominal joint velocity command is non-zero and points directly opposite to the safety boundary's outward normal vector, the safety filter cancels the command entirely, trapping the robot in a spurious equilibrium [9].

Theoretical Breakdown of the Trap

The complete cancellation of joint motion occurs due to two structural properties of the minimum-intervention safety filter :

1. Complete Normal Cancellation: Because the nominal goal command points straight into the obstacle, it violates the safety constraint requiring non-negative growth of the barrier function. To guarantee forward invariance of the safe set, the safety filter calculates the minimal correction necessary by subtracting the full magnitude of the inward velocity component. At the safety boundary, this subtraction equals one hundred percent of the nominal velocity vector, bringing forward motion to a complete halt.
2. Absence of Tangential Residual: Because a pure goal-attraction controller generates velocity directed solely toward the target, it contains no orthogonal or tangential component relative to the obstacle boundary. Once the safety filter removes the inward radial motion, no leftover velocity remains to slide or steer the manipulator sideways along the boundary.

As established by Reis et al. , these boundary deadlocks are not merely temporary slowdowns or unstable saddle points; they act as asymptotically stable attractors. The interaction between the goal field and the safety filter converts the boundary into a mathematical trap that actively attracts and holds the robot

state. Overcoming this structural limitation requires an explicit escape mechanism, such as a tangential circulation field, to break the opposing symmetry and guarantee continuous motion toward the target [1, 4, 9].

3.4.3. A Decoupled Escape in the Nominal Field

To break spurious equilibria without relaxing the safety constraint, an escape term is added directly to the nominal reference field $\dot{q}_{\text{nom}}$:

$$\tilde{\dot{q}}_{\text{nom}} = \dot{q}_{\text{nom}} + k_{\text{esc}} t(q) + k_{\text{out}} \frac{\nabla h}{\|\nabla h\|} \quad (44)$$

where $t(q)$ is a normalized unit vector orthogonal to the barrier gradient ($\nabla h(q)^T t(q) \approx 0$). Activation is governed by a hysteresis latch (engaging when $h(q) < h_{\text{engage}}$ and disengaging when $h(q) > h_{\text{release}}$), preventing chatter. The modified reference $\tilde{\dot{q}}_{\text{nom}}$ is then processed through the standard safety filter.

3.4.4. Safety Guarantee and Its Decoupling from the Escape

Injecting escape terms directly into the nominal velocity controller solves the deadlock problem while preserving the absolute safety guarantees provided by the Control Barrier Function.

Proposition 2 (Bias-Independent Forward Invariance): For any smooth barrier function, applying a bounded nominal velocity command—regardless of any added escape, circulation, or bias terms—guarantees that the filtered velocity command satisfies the safety constraint across the entire safe region. Consequently, the safe configuration set remains strictly forward invariant, meaning the robot is mathematically guaranteed never to enter an unsafe state, regardless of how the reference command is modified.

Core Principle of the Decoupled Guarantee

The fundamental strength of this formulation lies in the mathematical separation between command generation and safety filtering [1, 4]:

- **Projection Mechanics:** The safety filter acts as a geometric projection. It takes whatever nominal velocity command is requested by the high-level controller and projects it onto the set of allowable, safe joint velocities.
- **Boundary Integrity:** Adding an escape vector or tangential bias to the nominal reference shifts the input point being projected, but it never alters, shifts, or relaxes the underlying safety boundary itself.
- **Unconditional Safety:** Because the safety filter continuously enforces the barrier condition at the boundary, Nagumo's theorem guarantees that the system state can never cross into forbidden space, regardless of the direction or magnitude of the injected escape force [1, 4].

Key Structural Properties

1. **Reference Modification vs. Constraint Relaxation:** Rather than temporarily loosening safety thresholds to bypass an obstacle, this approach modifies only the reference trajectory. The strict safety envelope remains entirely intact, ensuring that collision avoidance is never compromised for the sake of mobility [10].
2. **Deadlock Elimination:** At configurations where a head-on deadlock would otherwise occur, the escape vector breaks the direct opposing alignment between the goal field and the safety gradient. By destroying this anti-collinear symmetry, the net velocity output remains non-zero, causing the robot state to slide and circulate smoothly along the safety boundary until the obstacle is cleared [10, 19].
3. **Stability Scope:** This decoupled design separates safety from goal seeking [10, 19]. It guarantees absolute whole-arm safety everywhere in the workspace, while preserving local asymptotic convergence toward the target whenever the robot is moving freely outside the filter's active intervention zone [10, 19].

3.5. Collision Avoidance Controller: Extension to the 3D Case and Circulation-Based Deadlock Avoidance

This section extends the planar formulation established in prior sections to the full three-dimensional, seven-degree-of-freedom (7-DOF) operational space of the Franka Emika FR3 manipulator [21]. Building directly upon the established joint-space obstacle mapping [11] and Control Barrier Function (CBF) filtering framework, this spatial extension addresses two key requirements:

1. The mathematical modifications necessary to map spatial obstacles and calculate barrier function gradients across a 7-DOF joint space.
2. The implementation of a continuous spatial circulation field to eliminate boundary deadlocks (spurious equilibria) [9, 10, 19].

The core notation, including obstacle center V , inflated radius R , link origins P_i , link lengths l_i , distances d_i , half-widths δ_i , critical headings Q_i^W , barrier function $h(q)$, nominal command v_{pid} , and CBF decay parameter λ , is retained directly from the planar formulation [11, 20].

Transitioning from the 3-DOF planar system to the spatial domain introduces three fundamental kinematic and structural modifications:

- **Configuration Space Expansion:** The active joint space expands from three planar joints to all seven active joints of the FR3 manipulator, defined by configuration vector $q \in \mathbb{R}^7$ [21].

- **Spatial Task and Obstacle Coordinates:** The controlled task transitions to the three-dimensional end-effector position $p_{ee}(q) \in \mathbb{R}^3$, while circular workspace obstacles generalize to spherical regions centered at $V \in \mathbb{R}^3$ with inflated safety radius R .
- **Kinematic Redundancy Resolution:** Controlling a 3D position task ($p_{ee} \in \mathbb{R}^3$) with a 7-DOF manipulator introduces four redundant degrees of freedom, requiring an explicit null-space projection scheme in the nominal command field [15, 21].

The structural framework of the obstacle model and the minimum-intervention CBF safety filter remains preserved; the primary mathematical modification inside the barrier function involves the calculation of spatial angular distance.

3.5.1. Per-Link Barrier in 3D

The forbidden-cone half-width δ_i defined in the planar formulation is constructed entirely from scalar geometric quantities: link length l_i , joint-to-obstacle workspace distance $d_i = \|V - P_i(q)\|$, and inflated obstacle radius R [11, 20].

Because these scalar distances are invariant under isometric spatial transformations, the closed-form expressions for δ_i (across geometric Cases A, B, and C) carry over directly to the three-dimensional domain without modification.

What changes fundamentally in $\mathbb{R}^3$ is the evaluation of the angular clearance. In the planar case, the orientation of link A_i and the line-of-sight to obstacle V are represented by scalar angles Q_i and Q_i^W , yielding a scalar offset $(Q_i - Q_i^W)$. In spatial operational space, link orientation and obstacle heading cannot be parameterized by single scalar angles; instead, they form unit directional vectors on the 2-sphere $S^2 \subset \mathbb{R}^3$ [24]:

$$\hat{u}_i(q) = \frac{P_{i+1}(q) - P_i(q)}{\|P_{i+1}(q) - P_i(q)\|}, \quad \hat{n}_i(q) = \frac{V - P_i(q)}{\|V - P_i(q)\|} \quad (45)$$

The exact spatial replacement for the planar angular difference $(Q_i - Q_i^W)$ is the unsigned geodesic angle $\phi_i \in [0, \pi]$ between these directional unit vectors [20, 24]:

$$\phi_i = \arccos(\hat{u}_i^T \hat{n}_i) \quad (46)$$

Substituting the spatial angle ϕ_i into the barrier definition yields the general 3D per-link Control Barrier Function:

$$h_i(q) = \phi_i^2 - \text{sgn}(\delta_i) \delta_i^2 \quad (47)$$

Exact Kinematic Equivalence

This vector formulation represents an exact geometric extension of the planar model rather than an approximation. The planar barrier function relies exclusively on the squared angular difference $(Q_i - Q_i^W)^2$. Because the absolute magnitude of the planar signed angle equals the unsigned inner-product angle ($|Q_i - Q_i^W| \equiv \phi_i$), the equality $(Q_i - Q_i^W)^2 = \phi_i^2$ holds identically [20].

The directional sign carried in the 2D formulation is rendered redundant by the quadratic power, establishing ϕ_i as the mathematically exact spatial generalization.

Empty Cone Refinement (Case C Extension)

In planar Case C—where link length l_i is insufficient to physically intersect the inflated obstacle boundary—the standard formulation sets $\delta_i = 0$, yielding $h_i(q) = \phi_i^2 \geq 0$. To enforce C^1 continuous transitions along the Case B/C boundary, δ_i is analytically continued into negative real values:

$$\delta_i = -\text{arccosh}\left(\frac{l_i^2 + d_i^2 - R^2}{2d_i l_i}\right) \quad (48)$$

Applying the signed-square operator in (47) ensures that an empty cone ($\delta_i < 0$) evaluates to:

$$h_i(q) = \phi_i^2 + |\delta_i|^2 > 0 \quad (49)$$

At the boundary separating Case B and Case C, the argument $\xi_i = \frac{l_i^2 + d_i^2 - R^2}{2d_i l_i} = 1$ yields $\delta_i = 0$ identically, preserving C^1 differentiability of $h_i(q)$ across workspace transitions. For non-empty cones (Cases A and B), $\text{sgn}(\delta_i) = +1$.

Handling Angular Singularities

The derivative of $\arccos(x)$ diverges as $x \rightarrow \pm 1$, corresponding to configurations where link A_i is collinear with the obstacle center ($\hat{u}_i = \pm \hat{n}_i$). While the planar signed angle avoided this singularity through smooth scalar subtraction, the 3D formulation handles the gradient unboundedness during numerical assembly via directional gradient norm-clipping [23], preventing numerical ill-conditioning while strictly preserving the deflection vector direction.

3.5.2. Whole-Arm Barrier Aggregation

Combining all 3D per-link barriers into a single whole-arm barrier $h(q)$ uses the soft-minimum operator [12, 18, 22], this is inherited from the 2D formulation:

$$h(q) = -\frac{1}{\beta} \ln \sum_{i=1}^N \exp(-\beta h_i(q)) \quad (50)$$

As in 2D, $h(q) \geq 0$ guarantees that all N links satisfy individual spatial safety limits.

3.5.3. Barrier Gradient in 3D

To enforce safety within the Control Barrier Function filter, the joint-space gradient of the aggregated whole-arm barrier, $\nabla h(q) \in \mathbb{R}^7$, must be evaluated at each control cycle. In the 2D planar model, each link contribution was evaluated analytically via scalar partial derivatives, $\frac{\partial h_i}{\partial Q_i} = 2(Q_i - Q_i^W)$, and combined into the whole-arm barrier gradient.

In spatial operational space, link A_i depends on configuration $q \in \mathbb{R}^7$ through the spatial forward kinematics of all upstream joints. Applying the chain rule to the soft-minimum (Log-Sum-Exp) barrier aggregation scheme [22] yields the full joint-space gradient:

$$\nabla h(q) = \sum_{i=1}^N \left(\frac{e^{-\beta h_i(q)}}{\sum_{j=1}^N e^{-\beta h_j(q)}} \right) \nabla h_i(q) \quad (51)$$

where N is the number of link constraints, $\beta > 0$ dictates the soft-minimum smoothing factor, and $\nabla h_i(q) \in \mathbb{R}^7$ represents the individual link barrier gradient.

Numerical Differentiation via Finite Differences

While $\nabla h_i(q)$ can be computed analytically by propagating spatial link Jacobians $J_i(q) \in \mathbb{R}^{3 \times 7}$, evaluating the aggregate gradient $\nabla h(q)$ via central finite differences provides a computationally efficient and numerically robust alternative [23]. Over the active seven joints of the manipulator, each k -th component of the gradient is computed as:

$$[\nabla h(q)]_k \approx \frac{h(q + \epsilon e_k) - h(q - \epsilon e_k)}{2\epsilon} \quad (52)$$

where $\epsilon \ll 1$ is the perturbation step size and e_k represents the k -th standard basis vector in $\mathbb{R}^7$. This central difference scheme provides second-order accurate gradient evaluation [23] while bypassing the explicit construction and multiplication of spatial link Jacobians across all 7-DOF.

Singularity Neutralization via Directional Norm Clipping

As established in Section 3.5.1, evaluating individual link barriers $h_i(q)$ involves the spatial inner-product angle $\phi_i = \arccos(\hat{u}_i^T \hat{n}_i)$. The derivative of $\arccos(x)$ contains a term proportional to $(1 - x^2)^{-1/2}$, which diverges as $x \rightarrow \pm 1$ when link A_i becomes collinear with the obstacle center line-of-sight vector ($\hat{u}_i = \pm \hat{n}_i$).

To prevent numerical solver ill-conditioning near these collinear singularities, the raw assembled gradient vector $g = \nabla h(q)$ is regularized using directional norm clipping:

$$g \leftarrow g \cdot \min \left(1, \frac{g_{\max}}{\|g\|} \right) \quad (53)$$

where $g_{\max} > 0$ defines the maximum allowable gradient norm threshold. This clipping operation bounds the magnitude of the joint-space gradient while strictly preserving its directional vector orientation, ensuring stable numerical optimization within the downstream safety filter.

3.5.4. The Safety Filter in 3D

The Control Barrier Function (CBF) filter in the three-dimensional operational space maintains the same minimum-intervention structure established in the planar formulation [1, 4]. Evaluated over the full 7-DOF configuration space, the time derivative of the aggregate barrier function is given by $\dot{h}(q) = g^T \dot{q}$, where $g = \nabla h(q) \in \mathbb{R}^7$ is the aggregate joint-space gradient evaluated via finite differences [23]. The high-order safety condition enforcing forward invariance of the safe set $\mathcal{S}$ is:

$$g^T \dot{q} \geq -\lambda h(q) \quad (54)$$

Given a nominal joint velocity command $v_{\text{nominal}} \in \mathbb{R}^7$, the minimum-intervention safety filter evaluates the active joint command $\dot{q}^* \in \mathbb{R}^7$ by solving the quadratic program (QP):

$$\dot{q}^* = \arg \min_{\dot{q} \in \mathbb{R}^7} \frac{1}{2} \|\dot{q} - v_{\text{nominal}}\|^2 \quad \text{s.t.} \quad g^T \dot{q} \geq -\lambda h(q) \quad (55)$$

Unconstrained Closed-Form Projection

When actuator physical bounds are unconstrained, the spatial QP admits an analytical closed-form solution representing an exact orthogonal projection onto the safe velocity half-space :

$$\dot{q}^* = v_{\text{nominal}} + \frac{\max(0, -\lambda h(q) - g^T v_{\text{nominal}})}{\|g\|^2} g \quad (56)$$

In unconstrained conditions, this active projection modifies the nominal command v_{nominal} strictly along the normalized direction of the gradient vector $g \in \mathbb{R}^7$ only when the nominal trajectory violates the safety condition ($g^T v_{\text{nominal}} < -\lambda h(q)$). When $h(q) > 0$ and the nominal command is strictly safe, the projection term evaluates to zero and $\dot{q}^* = v_{\text{nominal}}$ [1, 4].

Actuator Box Constraints and Forward Invariance

To ensure hardware compatibility with the Franka Emika FR3 manipulator [21], the filter incorporates physical actuator constraints directly into the optimization problem. Bounding the joint velocity, acceleration, and jerk limits simultaneously results in a state-dependent per-joint velocity box $\mathcal{U}(q, \dot{q}_{k-1}) = [\dot{q}_{\min}, \dot{q}_{\max}] \subset \mathbb{R}^7$:

$$\dot{q}^* = \arg \min_{\dot{q} \in \mathcal{U}(q, \dot{q}_{k-1})} \frac{1}{2} \|\dot{q} - v_{\text{nominal}}\|^2 \quad \text{s.t.} \quad g^T \dot{q} \geq -\lambda h(q) \quad (57)$$

This box-constrained QP retains high computational efficiency and is solved continuously in real time.

The theoretical forward-invariance guarantee ($\mathcal{S}$ remains invariant for all $t \geq 0$) inherited from the planar formulation holds strictly under the condition that the control command executed by the physical robot hardware matches the command certified by (54). Consequently, any low-level hardware safety governor or velocity limiter running downstream of the safety filter must be configured with bounds strictly looser than the state-dependent box $\mathcal{U}(q, \dot{q}_{k-1})$ to prevent uncertified command truncation and ensure preservation of formal safety proofs [5] (consistent with the execution integrity requirement in (24)).

3.5.5. Redundancy Solution

The operational end-effector velocity is mapped from joint velocities via the task Jacobian matrix $J(q) \in \mathbb{R}^{3 \times 7}$ according to $\dot{p}_{ee} = J(q)\dot{q}$. Controlling a 3D Cartesian position task with the 7-DOF Franka Emika FR3 manipulator [21] leaves a four-dimensional null space $\mathcal{N}(J) \subset \mathbb{R}^7$. This redundancy is resolved by projecting a secondary joint-space posture objective $v_{\text{posture}} \in \mathbb{R}^7$ into the null space, yielding a composite nominal command v_{nominal} that preserves primary task execution [2]:

$$v_{\text{nominal}} = J^\dagger v_p + (I_7 - J^+ J) v_{\text{posture}} \quad (58)$$

where $v_p \in \mathbb{R}^3$ denotes the task-space proportional control action, and $(I_7 - J^+ J)$ represents the orthogonal null-space projection matrix $N \in \mathbb{R}^{7 \times 7}$.

Dual Pseudoinverse Formulation and Task Decoupling

To guarantee both singularity robustness and exact task-space decoupling, the composite nominal controller v_{nominal} in (58) incorporates two distinct generalized inverses [15, 25] (extending the 2D task-space kinematic control scheme from (9) and (4):

- Task Damped Pseudoinverse ($J^\dagger$): The primary task velocity command is mapped using the damped least-squares (DLS) inverse $J^\dagger = J^T (J J^T + \lambda_d^2 I_3)^{-1}$, where $\lambda_d > 0$ is a state-dependent damping factor. This prevents numerical divergence and excessive joint speeds near kinematic singularities [2].
- Null-Space Moore-Penrose Inverse (J^+): The null-space projection matrix $N = I_7 - J^+ J$ is constructed exclusively using the unweighted Moore-Penrose pseudoinverse $J^+ = J^T (J J^T)^{-1}$ [2].

The mathematical distinction between $J^\dagger$ and J^+ is critical for maintaining task independence. Because the Moore-Penrose inverse satisfies $J J^+ = I_3$, the projector guarantees strict orthogonality: $J N = J (I_7 - J^+ J) = 0$. Conversely, the damped inverse does not satisfy this algebraic identity ($J (I_7 - J^\dagger J) \neq 0$). Constructing N with $J^\dagger$ would allow lower-priority posture motions to leak into operational space, corrupting primary end-effector tracking trajectories [25].

3.5.6. Circulation-Based Deadlock Avoidance

When executing target-reaching tasks where a workspace obstacle lies directly along the straight-line trajectory to the goal, purely reactive safety filters encounter local minima (spurious equilibria). While the 2-DOF planar controller addressed this through the addition of a discrete velocity term with the objective of deflecting the EE trajectory away from the obstacle [11], extending the task to the full 3D spatial domain requires a continuous mechanism to circumvent boundary deadlocks.

This section details the fundamental topological reasons behind reactive deadlocks and presents a smooth spatial circulation field designed to guarantee continuous motion around obstacles while strictly preserving Control Barrier Function (CBF) safety bounds [9].

3.5.6.1. Mechanism of 3D Local Minima

The Control Barrier Function (CBF) filter ensures safety by stripping away only the component of the nominal velocity vector v_{nominal} pushing into the obstacle, leaving motion along the safe boundary. A system local minima occurs when the nominal velocity points straight toward the obstacle center, making v_{nominal} parallel to the barrier gradient vector $g = \nabla h(q)$.

When v_{nominal} and g are collinear:

1. The component of v_{nominal} tangent to the obstacle surface is zero.
2. The safety filter cancels the forward velocity entirely to prevent collision .
3. The robot comes to a complete stop ($\dot{q}^* = 0$) in a spurious equilibrium state [9, 10].

This stall is an inherent property of reactive velocity projection [9, 12]. Moving from three joints to seven joints ($\mathbb{R}^3 \rightarrow \mathbb{R}^7$) does not eliminate these deadlocks: mathematically, projecting a vector parallel to g onto the space orthogonal to g always yields zero in any dimension $\mathbb{R}^n$ [12].

From a topological perspective, an obstacle creates an unreachable "hole" in the free workspace. Koditschek and Rimon [26] proved that no smooth, reactive feedback controller can steer a robot to its goal from every starting position without creating at least one unwanted equilibrium point. No gain tuning can eliminate this unwanted stable equilibrium point; it can only be moved or made unstable [9, 12].

The planar implementation bypassed this asymptotically stable attractor using a hard, logic-switched "escape" maneuver. However, discrete switching does not translate well to 3D spatial operations :

- Infinite Escape Routes: In 2D, the robot only chooses between two escape directions. In 3D space, the set of tangential escape directions is infinite [24].
- Control Chattering: Hard switching introduces sudden control jumps across decision boundaries. This causes actuator chattering and breaks the smooth mathematical guarantees needed for safe control [10].

To maintain smooth motion and formal safety guarantees, the discrete 2D switch is replaced in 3D by a continuous spatial circulation field [9, 10, 11, 19].

3.5.6.2. Construction of the Spatial Circulation Field

To smoothly break collinear alignments without introducing discrete switching chatter, a continuous tangential circulation velocity $v_{\text{circ}} \in \mathbb{R}^3$ is constructed in operational space. Let $p_{\text{ee}} \in \mathbb{R}^3$ denote the end-effector position, $V \in \mathbb{R}^3$ the obstacle center, and $r = p_{\text{ee}} - V \in \mathbb{R}^3$ the relative offset vector [10, 19, 24, 27].

Choosing a unit circulation axis $\hat{k} \in S^2$, the tangential swirl direction $t \in \mathbb{R}^3$ is generated via the normalized cross product and oriented toward the goal position p_{goal} [24]:

$$t = \frac{\hat{k} \times r}{\|\hat{k} \times r\|}, \quad t \leftarrow \text{sgn}(t^T(p_{\text{goal}} - p_{\text{ee}})) t \quad (59)$$

This orientation step guarantees that the circulation vector always assists motion along the half-space facing the goal rather than driving the robot away from its target.

Dual-Gate Modulation and Circulation Velocity

The overall circulation velocity magnitude is modulated by two smooth gating functions to ensure localized action and stable goal convergence [10, 19]:

$$v_{\text{circ}} = k_c \min\left(e^{-h/h_{\text{act}}}, w_{\text{max}}\right) \tanh\left(\frac{\|e\|}{e_0}\right) t \quad (60)$$

where $e = p_{\text{goal}} - p_{\text{ee}}$ represents the Cartesian tracking error, $k_c > 0$ is the circulation gain, $h_{\text{act}} > 0$ defines the barrier activation scale, $w_{\text{max}} > 0$ caps the maximum swirl weight, and $e_0 > 0$ is the convergence smoothing radius.

Operational-Space Integration and Gating Dynamics

The resulting circulation term v_{circ} (60) is added directly to the nominal task-space reference velocity before null-space projection (58) and inverse kinematics resolution [10, 24]:

$$\dot{p}_{\text{des, ee}} \leftarrow \dot{p}_{\text{des, ee}} + v_{\text{circ}} \quad (61)$$

The dual-gate structure guarantees smooth, chattering-free trajectory generation across the entire workspace:

- **Obstacle Proximity Gate** ($\min(e^{-h/h_{\text{act}}}, w_{\text{max}})$): Exponentially activates the circulation field only near the obstacle ($h \rightarrow 0^+$). Away from obstacles ($h \gg h_{\text{act}}$), $e^{-h/h_{\text{act}}} \rightarrow 0$, causing v_{circ} to vanish entirely and restoring unperturbed nominal control. The cap w_{max} prevents excessive tangential speeds near the boundary [19].
- **Goal Attenuation Gate** ($\tanh\left(\frac{\|e\|}{e_0}\right)$): Smoothly ramps down the circulation magnitude to zero as the end-effector approaches the target ($\|e\| \rightarrow 0$). This eliminates goal orbiting, prevents sign-test chattering near the target, and ensures exact asymptotic convergence to p_{goal} [19, 24].

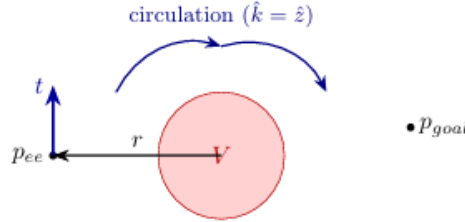


Figure 9: Representation of the circulation mechanism around the obstacle.

3.5.6.3. Efficacy and Safety Properties

1. **Deadlock Elimination:** v_{circ} provides tangential velocity at symmetric stalls ($\dot{q}_{\text{nom}} \parallel g$), transforming asymptotically stable attractor into repelling spirals [10, 19].
2. **Smooth & Chatter-Free:** v_{circ} is C^∞ continuous with respect to q , avoiding discrete switching thresholds.
3. **Transient Bounds:** Activation gates restrict circulation velocity to $\|v_{\text{circ}}\| \leq k_c w_{\text{max}}$.
4. **Invariant Safety:** v_{circ} enters only the nominal reference $\dot{q}_{\text{nom}}$; the CBF constraint ($g^T \dot{q}^* \geq -\lambda h$) remains unchanged, guaranteeing forward invariance of $\mathcal{S}$ under all operating conditions [10, 19].

3.5.6.4. Selection of Circulation Axis

The orientation of the unit circulation axis $\hat{k} \in S^2$ dictates the plane in 3D operational space within which the tangential velocity v_{circ} rotates [19, 24]. By adjusting $\hat{k}$, the system can adopt different obstacle bypass maneuvers suited to the workspace geometry:

- **Vertical Axis** ($\hat{k} = \hat{z}$): Generates a horizontal swirl in the xy -plane, directing the manipulator arm around the lateral side of the obstacle [10, 24]. This represents the default three-dimensional detour maneuver.
- **Obstacle-to-Goal Axis** ($\hat{k} = \frac{p_{\text{goal}} - V}{\|p_{\text{goal}} - V\|}$): Circulates motion within the plane orthogonal to the obstacle-to-goal line of sight, creating a minimal-distance spatial detour around the impediment [19, 24].
- **Horizontal Axis** ($\hat{k} \perp \hat{z}$): Rotates the circulation plane vertically, recovering the planar "over-the-top" avoidance trajectory characteristic of 2D models [11, 24].

Preservation of Forward Invariance

Crucially, injecting v_{circ} into v_{nominal} does not alter or compromise the formal safety guarantees established in Section 3.5.4. Because v_{circ} modifies only the unconstrained reference prior to QP optimization, the downstream 3D CBF filter in (57) continuously enforces the linear safety condition $g^\top \dot{q}^* \geq -\lambda h(q)$, preserving forward invariance of $\mathcal{S}$ identically to the 2D deadlock resolution in (21) [10, 19].

The downstream Control Barrier Function filter processes the resulting joint command and enforces the linear safety inequality $g^\top \dot{q} \geq -\lambda h(q)$ regardless of the direction or magnitude of v_{circ} . If the selected circulation axis drives the manipulator too close to a boundary, the CBF-QP filter dynamically trims the velocity component normal to the barrier, ensuring that forward invariance of the safe set $\mathcal{S}$ remains strictly intact across all choices of $\hat{k}$ [1, 4, 10].

Chapter 4 - Controller Architecture and Structure

In this chapter, an overview of how the controller is designed and structured will be presented. This includes the frameworks, operating systems, scripts, and programming languages used to develop a functional controller capable of sending specific commands to a 7-DOF robot, the Franka Emika FR3 , while fulfilling the objectives previously described in this thesis.

The controller is composed of several software components that work together rather than operating as a single independent program. Each component has a specific role, from communication between different processes to the execution of commands on the robot. This layered structure allows the high-level control logic developed in this thesis to remain separated from the low-level hardware communication required to operate the FR3 [21]. Understanding this architecture is therefore important for identifying where the proposed safety controller is integrated within the overall robotic system.

This chapter focuses on three main questions: what ROS 2, ROS2_control [13], and libfranka [21] are and what each component does; how they work together to create a control stack for the Franka arm [21]; and where the controller developed in this thesis, a high-level Control Barrier Function (CBF) safety filter that generates velocity commands, is positioned within this stack.

At a high level, the controller produces a desired velocity command based on the current state and objectives of the robot. This command is then passed through the ROS 2 and ros2_control [13] infrastructure before reaching the hardware interface responsible for communicating with the FR3 [21]. ROS 2 provides the communication framework between the different software components, while ros2_control [13] provides the structure for connecting controllers to robot hardware. Libfranka [21] then provides the interface required for communication with the Franka control system.

This separation is particularly useful for maintaining a modular and understandable control architecture. It allows the safety controller to focus on generating safe commands without directly handling the hardware-specific communication

with the robot. At the same time, the lower-level components can manage the timing and execution of these commands.

The software architecture described in this chapter serves as the execution framework for the mathematical formulation detailed in Chapters 2 and 3. The purpose of this section is to establish the software execution pipeline that carries the safe velocity commands—generated via nominal velocity synthesis (58), circulation fields (60), and the safety-filtering QP (57) from the high-level controller down to the physical robot, while providing an overview of ROS 2 and its role in modern robotic applications.

4.1. Control Pipeline overview

In this section the objective is to introduce the main parts of the controller pipeline that allow for the communication between user and the robot controller and the controller to the robots' actuators.

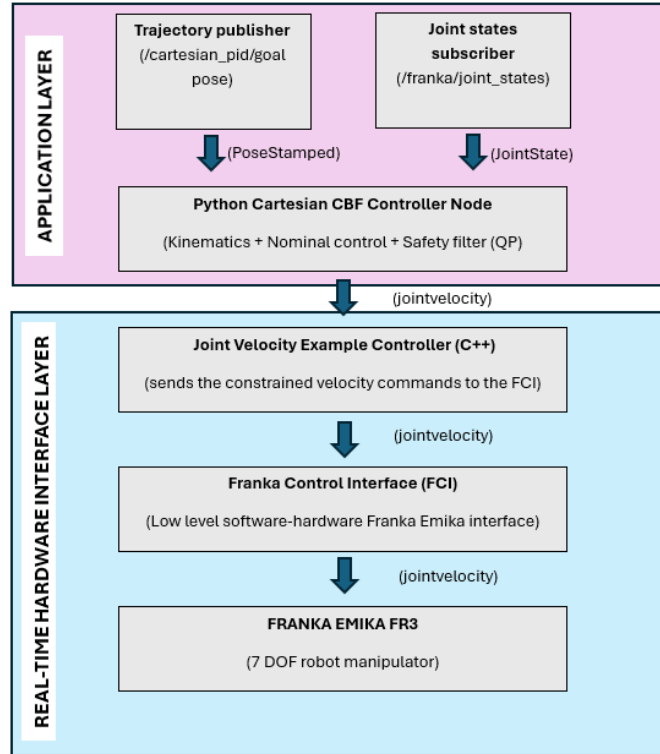


Figure 10: Full controller architecture scheme separating slow and fast update rates and each of the scripts in order of information flow.

The control architecture is divided into two primary domains: an Application Layer running at 100 Hz that executes high-level control logic, and a Real-Time Hardware Interface Layer operating at 1000 Hz that enforces low-level deterministic motor control [28]. Figure 10 illustrates the end-to-end signal flow through these layers.

4.1.1. What is ROS2?

ROS 2 is a modular, distributed middleware framework designed for robotic systems . Operating on top of an underlying operating system (typically Linux), it provides hardware abstraction, device drivers, package management, visual development tools, and a decentralized publish-subscribe communication architecture based on the Data Distribution Service (DDS) standard .

In other words, it is not an operating system in the traditional sense but a framework for building distributed robotic software out of independent processes, called nodes, that communicate over a publish/subscribe bus built on DDS. Each node can publish data on named topics, subscribe to topics published by other nodes, expose synchronous services, run longer-lived actions, and read configuration via parameters .

Then a launch script starts and configures a whole graph of nodes together [28, 29].

4.1.1.1. Nodes and communication mechanisms

The core building block of any ROS 2 application is the Node. A node is a single, purpose-built process dedicated to an isolated task, such as reading sensor data, driving a motor, or executing a path planning algorithm. All of this is included inside scripts; these can be programmed in C++ or Python [28].

Because each node runs independently, the system gains modularity, fault tolerance, and software reusability. A failure in one node does not inherently crash the rest of the robot, and individual components can be written, tested, and debugged in isolation.

Nodes need communication mechanisms to receive and send information to and from the robot or other nodes, creating thus a communication network that connects user and robot manipulator. The four main communication paradigms are as follows:

- **Topics:** Used for continuous, asynchronous data streaming. A node that produces data acts as a Publisher, sending typed messages over a named channel called a Topic. Nodes that require that data act as Subscribers, registering interest in the topic to receive incoming messages via callback functions. Publishers stream information and do not need to know who is receiving data, and subscribers don't need to know the origin of the data [28].

A simple analogy would be that of a radio broadcast and an anonymous listener, the publisher streams regardless of who is listening, and can have many listeners and the subscriber receives that information regardless of who is streaming it as long as it is listening to the same channel (in this case *topic*). This is ideal for streaming sensor data or other kinds of information that do not need a confirmation that has been received [28].

- **Services:** Used for synchronous, point-to-point request-and-response calls. A Service Server exposes an endpoint that performs a discrete computation or state change when triggered. A Service Client sends a request and waits for the server to process it and return a response. This paradigm is suited for fast, deterministic actions such as querying a sensor's current mode or setting a state variable. A simple analogy would be that of a phone call where the caller expects a response from the other side [28].
- **Actions:** Designed specifically for long-running, non-blocking robotic tasks. Actions extend the request-response model by using three underlying channels: a Goal channel to initiate and accept the task, a Feedback channel to continuously stream progress updates to the client while the task executes, and a result channel to transmit the outcome upon completion or cancellation. Actions are ideal for high-level tasks like mobile navigation or robotic arm trajectory execution [28].
- **Parameters:** Configuration settings maintained locally by individual nodes. Parameters allow developers to tune node settings (such as speed limits, physical constants, or filtering thresholds) at runtime without needing to recompile the source code. These can be goal positions, velocities, efforts to send to the joints and allow to customize the tasks while keeping a reusable and general source code [28].

For this thesis most of the communication tasks have been carried out using topics and servers as they are the simpler forms of data transfer in ROS2 and for straightforward tasks such as sending and receiving information to and from the Franka Emika FR3 they handle it competently.

4.1.1.2. Launch files

ROS 2 Launch Files are system orchestration scripts designed to automate the setup, configuration, and execution of a robot's computation graph. Instead of launching dozens of individual ROS 2 nodes manually, a launch file allows developers to start multiple independent processes across local or distributed networks using a single command. Written primarily in Python (or declaratively in XML/YAML), launch scripts manage node lifecycles, pass configuration variables, set environment flags, and conditionally trigger process execution [28,29].

They also allow dynamic topic and service remapping at startup, enabling identical node code to run across different hardware configurations without

recompilation. By nesting launch files within one another, complex robotic systems maintain a modular, scalable, and fully automated deployment process [28,29].

4.1.2. What is ROS2control?

While ROS 2 provides the messaging layer that lets nodes exchange data [28], it says nothing on its own about how to actually drive a motor at a guaranteed, fixed rate [13], that is the job of ROS2_control [13], a framework built on top of ROS 2 that standardises the interface between a controller and a piece of hardware, regardless of which robot is attached [13]. It is organised around three pieces: a hardware interface, a plugin specific to the robot that exposes named, readable state interfaces (joint position, joint velocity) and writable command interfaces (joint velocity command); a controller manager, a single process that owns the real-time loop and, on every tick, reads the hardware state, runs every active controller's update step, and writes the resulting commands back out; and controllers themselves, the plugins that implement that update step [13].

A simple analogy would be that of a factory assembly line with a fixed conveyor-belt speed: whatever is placed on the belt each cycle, in this case, a velocity command, is guaranteed to reach the next station on time, regardless of what produced it.

For this thesis, ROS2_control [13] is the layer that receives the safety-filtered velocity published by the CBF controller and turns it into an actual, rate-guaranteed write to the FR3's [21] joints, through the JointVelocityExampleController mentioned in section 4.1 [13]: it reconstructs the 100 Hz command stream into a jerk-limited signal at the controller manager's much faster loop rate (1000 Hz) [2, 13], and falls back to a safe, ramped-down stop if that stream ever goes quiet [13]. Its usefulness to this thesis is precisely its indifference to why a velocity was chosen, it lets the CBF safety filter be developed, tuned, and reasoned about entirely independently of the real-time hardware-driving code underneath it [13].

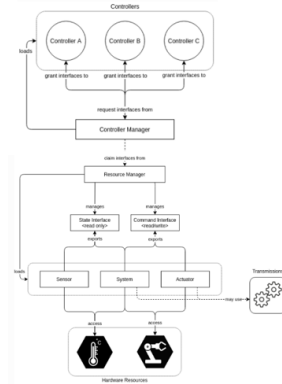


Figure 11: ROS2 control structural scheme in order of information flow.

4.1.3. What is libfranka?

Libfranka is Franka Robotics' own C++ library for direct, low-level access to the robot over the Franka Control Interface (FCI) , a dedicated real-time link to the arm's control unit running at 1 kHz with a hard per-cycle deadline . It supports several control modes (joint position, joint velocity, joint torque, Cartesian velocity/pose) , and beneath whatever command it is given, the robot's own firmware enforces its own hard joint position, velocity, acceleration, and jerk limits, a safety backstop that exists completely independently of anything computed on the ROS 2 side [21].

On its own, libfranka does not take into account ROS 2 [28] or ROS2_control [13]; the bridge is franka__hardware, part of the franka__ROS2 project , which wraps libfranka [21] as a standard ROS2_control [13] hardware interface, so that from the controller manager's point of view the FR3 looks like any other piece of ROS2_control-compliant [13] hardware[13 , 21].

For this thesis, libfranka (via franka__hardware) is what makes the rest of the architecture possible at all: it is the only component that actually talks to the physical robot, carrying the jerk-limited velocity command from JointVelocityExampleController across the FCI at 1 kHz , and its firmware-level limits act as a final, independent safety net underneath the CBF-based guarantee [1, 2 ,4 ,21].

It does not guarantee safety in terms of obstacle avoidance but in terms of position, velocity, acceleration and jerk limits considering the robot's limitations, in other words it guarantees only firmware safety, software safety is guaranteed at a higher level. It acts as the last line of defence if, for any reason, an unsafe command was ever to reach that boundary [21].

4.2. Controller pipeline components summary

To complement the architectural overview in Figure 4.1, the specific functional role of each software and hardware component within the control pipeline is summarized below:

1. **Trajectory Publisher (/cartesian_pid/goal_pose):** Serves as the high-level goal generator. It defines the desired end-effector target poses in Cartesian space and publishes them as PoseStamped messages to the controller .
2. **Joint States Subscriber (/franka/joint_states):** Handles state estimation feedback at the application layer. It receives raw FrankaJointState data coming from the hardware layer, formats it into standard JointState messages, and feeds it into the CBF controller to close the feedback loop.
3. **Python Cartesian CBF Controller (Application Layer):** Represents the primary algorithmic contribution of this thesis. Running at 100 Hz, it processes target goal poses and current joint states to compute nominal operational-space velocities (58), evaluates per-link geodesic barriers (47) and soft-min aggregated constraints (50), and solves the 3D safety-filtering Quadratic Program (55) under state-dependent velocity, acceleration, and jerk limits $\mathcal{U}(q, \dot{q}_{k-1})$ (57). The resulting safe joint velocity vector $\dot{q}^*$ is published onto the /jointvelocity topic.
4. **Joint Velocity Example Controller (Real-Time Hardware Interface Layer):** Functions as the C++ bridge between high-level logic and low-level hardware execution. Running inside ROS2_control at 1000 Hz, it receives the 100 Hz velocity commands $\dot{q}^*$ (57) applies smoothing and jerk limitations, and ensures deterministic delivery to the hardware interface without violating execution integrity (**Section 3.5.4**).
5. **Franka Control Interface / libfranka (FCI):** Acts as the proprietary low-level interface provided by Franka Robotics. It transmits 1000 Hz velocity commands over a dedicated real-time link to the arm's control unit while enforcing native firmware limits (position, velocity, acceleration, and jerk) as an independent hardware safety net [21].
6. **Franka Emika FR3:** The physical 7-DOF robot manipulator that executes the requested motion in the real world. It continuously measures its internal sensors and streams real-time state feedback (FrankaJointState) back up to the application layer at 1000 Hz [21].

4.3. Controller architecture implementation in ROS2

In this thesis, the concepts described above come together by connecting three different scripts (each one with a distinct function) that never call each other directly but instead pass messages back and forth the way a radio station and its listeners do [28].

The safety controller developed in this work, the Python Cartesian CBF Controller, operates as follows: it listens on two channels—one carrying target goal poses (`/cartesian_pid/goal_pose`) and one carrying current joint states (`/franka/joint_states`). Roughly one hundred times per second (100 Hz), it calculates a safe joint velocity command $\dot{q}^*$ and broadcasts that value on a third channel (`/jointvelocity`). These safety criteria enforce joint boundaries and obstacle clearance via Control Barrier Functions (51) - (54), alongside physical motor acceleration and jerk constraints using state-dependent feasibility sets $\mathcal{U}(q, \dot{q}_{k-1})$ (57).

It does not send this value directly to the motors, nor does it track which specific process is listening; it simply publishes the data over the ROS 2 topic bus [28].

A second program, Joint Velocity Example Controller (C++), running as part of the `ros2_control` framework [13] and built specifically for the Franka arm [21], is tuned in to that same channel [28]; its only job is to take whatever velocity value arrives and turn it into a smooth, continuously updated signal that can be sent to the robot's joints many times faster (1000 Hz) than the safety controller itself runs, while also bringing the robot safely to a stop if that channel ever goes quiet [13].

This program, in turn, does not talk to the robot directly either: it hands its output to a third piece of software, the FCI, built on Franka's own `libfranka` library [21], whose sole responsibility is to carry that signal across the dedicated, high-speed link to the robot's control unit, where the robot's own built-in limits provide one final layer of protection before any motion actually happens. A Service call is used only once in this whole chain, at start-up, when the low-level controller needs to ask another part of the system for a description of the robot it is controlling (`/FrankaJointState`), a single question with a single answer, unlike the continuous streaming used everywhere else [21, 28].

The result of this arrangement is that each program can be understood, tested, and trusted on its own: the high-level safety bounded controller only has to be concerned with choosing a safe velocity, and everything below it is responsible for reliably and safely getting that velocity to the physical robot, without either side considering the internal details of the other [13, 28].

Chapter 5 - Controller safety layer design

While Chapters 2 and 3 established the continuous-time control theory behind the 3D Control Barrier Function (CBF) [1, 4], deploying these algorithms on a physical 7-DOF manipulator requires bridging the gap between theoretical models and real-time code [13, 21]. Continuous equations assume instantaneous actuation and perfect feedback, but real hardware operates under discrete sampling clocks, communication latencies, and hard motor limits.

This chapter aims to detail the software design, control architecture, and hardware implementation of the safety-filter script that proposes a safe velocity to the lower-level controllers for the Franka Emika FR3 . The architecture divides responsibilities across a multi-sampling rate hierarchy: a high-level ROS 2 Python node running at 100 Hz computes robot dynamics using Pinocchio, resolves differential kinematics, and solves a custom scalar-dual Quadratic Program (QP) to guarantee collision avoidance [1]. Lower down, the Franka Control Interface (FCI) receives these certified velocity setpoints and executes joint-level tracking at 1 kHz [13, 21].

Furthermore, the script where the most relevant results are obtained and plotted in order to verify the performance of the controller during the task execution is also detailed in section 5.6.

The purpose of each function written in the script is detailed as follows:

Table 2: Scripts and their functions inside the safety layer and controller.

Controller Subsystem	Python Implementation Component	Mathematical / Functional Role
Kinematic Model	<code>pinocchio.buildModelFr</code>	Calculates forward kinematics and frame Jacobians.
Angular Barrier	<code>cap_angle(delta)</code> & <code>link_barrier()</code>	Evaluates unsigned 3D angle ϕ_i and link distance ρ_i for segments $\in [3, 8]$.
Softmin Aggregation	<code>soft_min()</code>	Generates a single C^∞ -smooth underestimate $h(q)$ across all link pairs.
Gradient Estimator	<code>h_and_grad()</code>	Approximates $\nabla h(q)$ via finite differences with singularity magnitude clipping.
Circulation Field	<code>circulation_velocity()</code>	Generates a tangent vector $\mathbf{v}_{\text{circ}}$ to guide the end-effector around obstacles.
Redundancy Resolution	<code>control_loop()</code> (Task/Null-space)	Damped Least-Squares (DLS) for task space; exact pseudo-inverse $N(q)$ for posture.
Kinematic Bounds	<code>actuator_box()</code>	Computes velocity limits constrained simultaneously by velocity, acceleration, and jerk.
Safety Filter	<code>safety_qp()</code>	Solves a 1D dual QP for optimal modification δ subject to $h(q + \delta) \geq 0$.

5.1. Kinematic Model & Robot Representation

The controller safety layer relies on an accurate real-time representation of the Franka Emika FR3 manipulator's kinematic structure [2, 21]. Rigid-body kinematic evaluations are offloaded to the C++ core of the Pinocchio spatial robotics library, enabling execution within the 100 Hz control loop [2, 28].

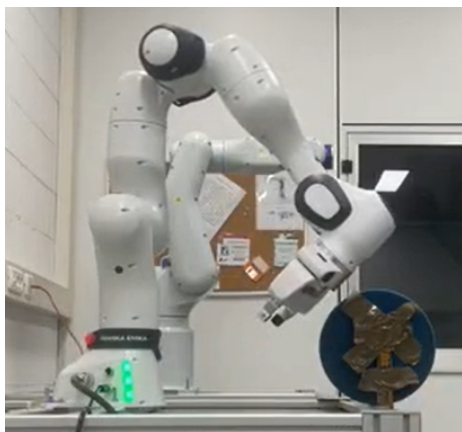


Figure 12: Physical FR3 at the intelligent robotics lab.

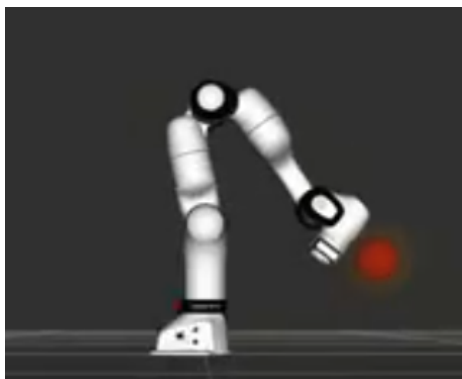


Figure 13: Real-time representation in RVIZ of the FR3 .

5.1.1. Model Parsing & Frame Assignment

During node initialization, the manipulator's Unified Robot Description Format (URDF) file is parsed into a kinematic tree [14]. The model establishes transformation matrices $T_i^0(q)$ mapping each joint frame i back to the base frame origin [2].

To protect the full physical volume of the arm rather than relying on a simplified point-mass abstraction, the algorithm monitors five sequential link segments:

$$\mathcal{S}_{\text{links}} = \{(\text{link3}, \text{link4}), (\text{link4}, \text{link5}), (\text{link5}, \text{link6}), (\text{link6}, \text{link7}), (\text{link7}, \text{link8})\}$$

Each pair $(\text{base}_i, \text{end}_i)$ defines a bounded spatial cylinder representing the physical geometry of link i .

5.1.2. Forward Kinematics & Task Jacobian Extraction

At each control tick (k) , received joint positions $q \in \mathbb{R}^7$ trigger a forward kinematics pass to update operational pose $x = [p^T, \phi^T]^T \in \mathbb{R}^6$ via $T_e^b(q) \in SE(3)$ (1). The spatial frame Jacobian $J_{\text{spatial}} \in \mathbb{R}^{6 \times 7}$ maps joint rates to Cartesian velocity according to differential kinematics (3):

The translational task Jacobian $J \in \mathbb{R}^{3 \times 7}$ is isolated by extracting the top three rows of $J_{\text{spatial}}(q)$. This mapping relates joint-space velocities directly to Cartesian linear velocities $\dot{p}_{ee} \in \mathbb{R}^3$ of the end-effector, providing the geometric foundation for operational tracking error $e(t) = x_d(t) - x(q(t))$ (2) and subsequent null-space in Equation (7) [2].

5.2. 3D Capisani Angular Barrier & Softmin Aggregation

The system computes an angular Control Barrier Function (CBF) across all link segments to evaluate collision risk in 3D space.

5.2.1. Spatial Link Geometry

For each link segment $i \in \mathcal{S}_{\text{links}}$ defined by base position $p_{\text{base},i}$ and end position $p_{\text{end},i}$, the directional link vector u_i and the displacement vector toward the obstacle center d_i are given by [11]:

$$u_i = p_{\text{end},i} - p_{\text{base},i}, \quad d_i = p_{\text{obs}} - p_{\text{base},i} \quad (62)$$

The unsigned 3D angle $\phi_i \in [0, \pi]$ between these vectors is:

$$\phi_i = \arccos \left(\text{clip} \left(\frac{u_i \cdot d_i}{\|u_i\| \|d_i\|}, -1.0, 1.0 \right) \right) \quad (63)$$

The angular boundary δ_i defines the safe angular half-width around the obstacle of effective radius $r_{\text{eff}} = r_{\text{obs}} + r_{\text{buffer}}$, this parameter is calculated from (28) found in Chapter 3:

The individual link barrier value is defined as:

$$h_i(q) = \phi_i^2 - \text{sign}(\delta_i) \delta_i^2 \quad (64)$$

Squaring ϕ_i eliminates directional sign ambiguities in 3D workspace configurations [1, 24].

The equations mentioned in this subsection have been adapted to match the script nomenclature but are considered equivalent for convenience.

5.2.2. Softmin Aggregation & Gradient Estimation

To maintain differentiability across multiple spatial link barriers $h_i(q) = \phi_i^2 - \text{sgn}(\delta_i) \delta_i^2$ (64) without non-smooth switching, individual link values are unified using whole-arm soft-minimum log-sum-exp aggregation $h(q)$ with smoothing parameter β :

$$h(q) = -\frac{1}{\beta} \ln \sum_{i=1}^N \exp(-\beta h_i(q)) \quad (65)$$

The joint-space gradient $g = \nabla h(q)$ is approximated via central finite differences with perturbation step ϵ :

$$[\nabla h(q)]_k \approx \frac{h(q + \epsilon e_k) - h(q - \epsilon e_k)}{2\epsilon} \quad (66)$$

Near anti-parallel alignments ($\phi_i \rightarrow \pi$), high gradient magnitudes are regularized via directional norm clipping:

$$g \leftarrow g \cdot \min\left(1, \frac{g_{\max}}{\|g\|}\right) \quad (67)$$

5.3. Goal-Attenuated Circulation Vector Field

Standard CBF-QP safety filters can induce deadlock conditions (spurious equilibria) when nominal joint velocity opposes the barrier gradient ($\dot{q}_{nom} \parallel -\nabla h(q)$). To guarantee deadlock mitigation, a spatial tangential circulation field v_{circ} is constructed using normalized swirl vector $t = \frac{\hat{k} \times r}{\|\hat{k} \times r\|}$ oriented toward the target position ($t \leftarrow \text{sgn}(t^T(p_{goal} - p_{ee})) t$) . [6,9,26]

The circulation velocity magnitude is governed by the dual-gated circulation formulation (68):

$$v_{circ} = k_c \min\left(\exp\left(-\frac{h}{h_{act}}\right), w_{\max}\right) \tanh\left(\frac{\|e\|}{e_0}\right) t \quad (68)$$

This circulation velocity is injected directly into the nominal operational-space reference before differential kinematics resolution (54):

$$\dot{p}_{des,ee} \leftarrow \dot{p}_{des,ee} + v_{circ} \quad (69)$$

yielding modified task velocity $v_{task} = (\dot{x}_d + K_p e) + v_{circ}$

5.3.1. Swirl Vector Generation

Given an alignment axis $a = [0, 0, 1]^T$ (z-axis alignment), the baseline tangential swirl vector around the obstacle center x_{obs} is computed as [27]:

$$v_{\text{swirl}} = a \times (x_{\text{ee}} - x_{\text{obs}}) \quad (70)$$

The vector is normalized ($v_{\text{swirl}} \leftarrow v_{\text{swirl}} / \|v_{\text{swirl}}\|$) and its direction is flipped if it opposes the path to the target position x_{goal} :

$$\text{if } v_{\text{swirl}} \cdot (x_{\text{goal}} - x_{\text{ee}}) < 0 \implies v_{\text{swirl}} \leftarrow -v_{\text{swirl}} \quad (71)$$

5.3.2. Barrier Proximity & Goal Attenuation Gating

The circulation speed is scaled by barrier proximity and deactivated smoothly near the target to prevent orbiting [27]:

Barrier Gate: Activates when close to the barrier boundary ($h_{\text{act}} = 0.05$):

$$w(h) = k_{\text{circ}} \cdot \min \left(\exp \left(-\frac{h}{h_{\text{act}}} \right), w_{\text{cap}} \right) \leftarrow -v_{\text{swirl}} \quad (72)$$

Goal Attenuation Gate: Fades circulation smoothly to zero as tracking error $e = x_{\text{goal}} - x_{\text{ee}}$ approaches zero ($e_0 = 0.06$ m):

$$\gamma_{\text{goal}}(e) = \tanh \left(\frac{\|e\|}{e_0} \right) \quad (73)$$

The modified task velocity combines nominal proportional feedback v_p and the gated circulation field:

$$v_{\text{task}} = v_p + \gamma_{\text{goal}}(e) \cdot w(h) \cdot v_{\text{swirl}} \quad (74)$$

Because this modification affects only the nominal controller prior to filtering, safety guarantees remain strictly governed by the downstream QP safety filter [1].

5.4. Resolved-Rate Kinematics & Null-Space Projection

Converting task-space velocities $v_{\text{task}} \in \mathbb{R}^3$ to nominal joint velocities $\dot{q}_{\text{nom}} \in \mathbb{R}^7$ requires handling redundant degrees of freedom without introducing unwanted task-space drift [2,25].

5.4.1. Damped Task Inversion & Pure Null-Space Projection

Primary task tracking near kinematic singularities utilizes the task damped pseudoinverse $J^\dagger$ with regularization factor :

$$J^\dagger = J^T (JJ^T + \lambda_d^2 I_3)^{-1} \quad (75)$$

To enforce strict orthogonality ($JP_N = 0$) and prevent secondary task leakage into operational space, the null-space projection operator $P_N(q) \in \mathbb{R}^{7 \times 7}$ relies on the un-damped Moore-Penrose pseudoinverse $J^+ = J^T (JJ^T)^{-1}$ (7):

$$P_N(q) = I_{7 \times 7} - J^+(q)J(q) \quad (76)$$

Integrating secondary rest-posture bias $\dot{q}_0 = K_{\text{null}}(q_{\text{rest}} - q)$, the composite nominal joint velocity $\dot{q}_{\text{nom}}$ is synthesized as in Equation (77) [25]:

$$\dot{q}_{\text{nom}} = J^\dagger(q)v_{\text{task}} + P_N(q)\dot{q}_0 \quad (77)$$

5.4.2. Nominal Velocity Composition

The combined nominal joint velocity pulls redundant joints toward a preferred rest configuration :

$$\dot{q}_{\text{nom}} = J_{\text{damped}}^\dagger v_{\text{task}} + N(q)(-k_{\text{post}}(q - q_{\text{rest}})) \quad (78)$$

5.5. Unified Kinematic Bounding & Dual QP Safety Filter

The final control output is certified by projecting $\dot{q}_{\text{nom}}$ onto the safe control set defined by joint limits and the Control Barrier Function [4].

5.5.1. Dynamic Actuator Bounds

Joint velocity ($\dot{q}_{\text{max}}$), acceleration ($\ddot{q}_{\text{max}}$), and jerk (j_{max}) bounds are combined into a single dynamic velocity interval $[lb, ub]$ at each time step [5]:

$$\ddot{q}_{\text{prev}} = \text{clip} \left(\frac{\dot{q}_{k-1} - \dot{q}_{k-2}}{\Delta t}, -\ddot{q}_{\text{max}}, \ddot{q}_{\text{max}} \right) \quad (79)$$

$$a_{\text{lo}} = \max(-\ddot{q}_{\text{max}}, \ddot{q}_{\text{prev}} - j_{\text{max}}\Delta t), \quad a_{\text{hi}} = \min(\ddot{q}_{\text{max}}, \ddot{q}_{\text{prev}} + j_{\text{max}}\Delta t) \quad (80)$$

$$lb = \max(-\dot{q}_{\text{max}}, \dot{q}_{k-1} + a_{\text{lo}}\Delta t), \quad ub = \min(\dot{q}_{\text{max}}, \dot{q}_{k-1} + a_{\text{hi}}\Delta t) \quad (81)$$

5.5.2. Closed-Form Scalar Dual QP

The certified joint velocity command $\dot{q}^*$ minimizes the deviation from $\dot{q}_{\text{nom}}$ while enforcing the linear barrier condition $g^T \dot{q} \geq -\lambda_{\text{cbf}} h$:

$$\min_{\dot{q} \in \mathbb{R}^7} \frac{1}{2} \|\dot{q} - \dot{q}_{\text{nom}}\|^2 \quad \text{s.t.} \quad g^T \dot{q} \geq -\lambda_{\text{cbf}} h, \quad lb \leq \dot{q} \leq ub \quad (82)$$

Because the inequality constraint is scalar, the optimal joint velocity can be parameterized by a single dual scalar variable $\mu \geq 0$ [4]:

$$\dot{q}(\mu) = \text{clip}(\dot{q}_{\text{nom}} + \mu g, lb, ub) \quad (83)$$

The optimal multiplier μ^* is solved using a two-phase line search and bisection algorithm. The resulting command is smoothed using an exponential low-pass filter ($\alpha = 0.3$) before being published to the robot controller via the `ros2_control` bus [21]:

$$\dot{q}_{\text{pub}} = \alpha \dot{q}^* + (1 - \alpha) \dot{q}_{\text{filt}, k-1} \quad (84)$$

5.6. Offline Diagnostic & Validation Pipeline (plot_results.py)

The plot_results.py script processes real-time controller telemetry logs offline to evaluate safety compliance, motion quality, and controller overhead across four primary diagnostic figures.

5.6.1. Validation & Diagnostic Figures

- **Figure 1: Safety Manifold & Filter Compliance:**

Overlays the softmin barrier $h(q)$ with the true minimum link distance $\min_i h_i$ to distinguish mathematical softmin bias $(-\ln N/\beta)$ from physical collisions. It tracks the certified constraint margin $\text{cstr} = g^T \dot{q} + \lambda h$ to identify low-level hardware driver execution gaps ($\text{cstr} \geq 0$ while $\min_i h_i < 0$), alongside active escape/circulation states.

- **Figure 2: Generalized Joint Kinematics:**

Plots joint positions (q), filtered velocity commands ($\dot{q}$), and derived numerical accelerations ($\ddot{q}$) across all active joints to confirm compliance with actuator limits.

- **Figure 3: Workspace Trajectory Projections:**

Displays dual spatial projections—top-down X - Y swirl plane view to verify tangential circulation around obstacle bounds ($r_{\text{eff}} = R + \text{buffer}$) and side X - Z elevation views.

- **Figure 4: Cost of Safety (Temporal Efficiency & Energy Effort):**

Evaluates controller overhead by tracking trajectory completion percentage (0% to 100%) against elapsed time, highlighting intervals where active safety filtering is engaged. It integrates cumulative mechanical work ($E_{\text{cum}} = \int |\tau^T \dot{q}| dt$) to quantify the exact energy and time delay penalties introduced by safety constraints.

5.6.2. Terminal Verdict Logging

Upon execution, the pipeline outputs an automated safety verdict:

1. **Safety Boundary Compliance:** Confirms whether $\min_i h_i \geq 0$ was maintained throughout the trajectory .
2. **Execution Gap Analysis:** Calculates the percentage of sub-zero barrier ticks attributable to low-level actuator lag rather than safety filter failure.

The lines of code regarding the terminal verdict logging simply print out a pass/fail verification in the user terminal in order to check if the safety constraints have been respected and the robot manipulator has completed the task; it has mainly a diagnostic function for the code development.

All the code and scripts mentioned in this chapter are available in the `Robotics_PYTHON_scripts` GitHub repository, specifically in the `CBF_controller_3D/atmain` directory.

Chapter 6 - Results and Discussion

In this section the validation via simulation, data collection and experimental testing with the real Franka Emika FR3 will be presented. The aim of this chapter is to verify that the controller respects the h boundary of the CBF while completing the tasks in the 2D and 3D space. The discrete data is obtained from the `/joint_states` topics, from the ROS2 environment, that are published when launching the controller framework.

The `/joint_states` topic broadcasts the real-time kinematic and dynamic feedback of the robot's actuators. It maps each uniquely named joint to its current physical position (in radians or meters), velocity (in rad/s or m/s), and active effort/torque (in $\text{N} \cdot \text{m}$ or N), bound together by a high-precision hardware timestamp.

6.1. Methodology

Two main verification methods have been chosen:

RVIZ visualization and ROS2 topics publication (*/joint_states*) via which we can obtain the discrete data from the robot’s joint state to then present the vital performance metrics in graph form.

Validation using the real robot in the lab and obtaining the discrete data from the */franka_joint_states* topic, which publishes the current state of the actual physical robot’s joints.

To run the ROS2 nodes, scripts and controllers the computer operating the robot operating system used is ROS2 Humble Hawksbill 22.04 on Ubuntu 22.04 LTS with the *ROS2_control* framework, *pandas*, *numpy*, *Pinocchio* and *franka_hardware* libraries.

To test the controller performance two different experiment settings have been set up, one in 2D, where the only actuated joints are joints 1,3 and 5 to keep the robot fixed in the XZ plane and a different experiment setting where all joints are actuated and the robot is free to move in the 3D space without further constraints than the collision avoidance logic.

Additionally, using the second experiment setting in the 3D case, a virtual robot in the RVIZ simulation environment, a three-point trajectory has been imposed in which the “cost of safety” ,in terms of time and energy, has been evaluated by comparing how expensive the safety features of the controller are compared to a nominal controller.

For both experiment settings an obstacle of a predetermined diameter has been set in the reachable workspace of the robot. Then, after proper robot setup, the robot has been commanded into a trajectory that intersects with the object.

The expected result from these experiment settings is that the robot will move its geometry around the obstacle to reach the final commanded position.

Furthermore, in the case of finding a local minimum in its path, instead of staying still, the escape latch of circulation vortex logic will take over to make it slide along the obstacle radius surface without intersecting it and eventually reaching its final pose. This task will be performed in a manner such that the $h(t)$ barrier value stays positive, which in turn means, that the robot trajectory respects the obstacle radius and the additional safety buffer of 0.025m and does not collide.

6.1.1. 2D experiment setup:

For this experiment the only actuated joints are 1,3 and 5 (considering the FR3 nomenclature which states the base joint as joint 0) to constrain the robot manipulator motion to the XZ plane. A planar circular obstacle of radius 6cm is then placed in the coordinates (0.45, 0.25) of the cartesian space. This position intersects with the robot manipulator's dexterous workspace, which means it's a position where the robot's EE could be placed in a multitude of poses.

By placing it in this position it is sure that the obstacle obstructs potential trajectories of the manipulator.

6.1.1.1. RVIZ 2D experiment setup

For this virtual experiment the `/tf` and `/joint_states` topics are published for RVIZ to read them, this way the FR3 base is placed onto the (0,0,0) coordinate or the virtual space. Then the obstacle is also published and placed onto coordinates (0.45, 0, 0.25).

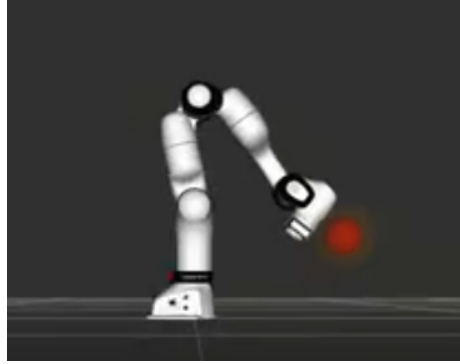


Figure 14: RVIZ representation of the FR3 and obstacle in the testing position.

The robot is then ordered to move its EE from cartesian position (0.30, 0, 0.25) to position (0.60, 0, 0.25) forcing thus a trajectory that would collide against the obstacle if not for the collision avoidance features of the controller.

The objective of this experiment is to observe the behaviour and performance of the controller when having to perform a task while also avoiding an obstacle.

Table 3: Code parameters used during the virtual testing in the 2D case using RVIZ.

Parameter 2D RVIZ	Value	Meaning / Units
dt	0.01	Control loop period (s, 100 Hz)
k_{task}	1	Task-space velocity
$v_{\max task}$	0.25	Max task-space speed cap (m/s)
λ_{cbf}	15	CBF class- K gain (conservativeness of barrier response)
β	15	Soft-min sharpness for blending per-link barriers
$grad_{clip}$	40	Max allowed norm of ∇h before rescaling
k_{esc}	0.15	Tangential escape velocity gain
$cstr_{release}$	0.2	Hysteresis release threshold for escape latch
n_{iter}	60	Bisection iterations for QP root-find
ϵ	1.00×10^{-4}	Central-difference step for barrier gradient

6.1.1.2. Franka Emika FR3 lab 2D experiment setup

For the lab experiment with the FR3 robot the position of its base is considered as the (0,0,0) coordinate. The obstacle is placed on the same coordinates as in the previous experiment using the virtual environment. This way the results can be directly compared.

The robot is then ordered to move its EE from the cartesian coordinates (0.30, 0, 0.25) to (0.60, 0, 0.25) forcing thus a trajectory that would collide against the obstacle if not for the collision avoidance features of the controller. The obstacle is modelled using cardboard and duct tape to simulate the sphere in 2D in the XZ plane, the choice of materials is obvious since with these materials the parts can be made and assembled within minutes and in case of collision the equipment is not damaged.

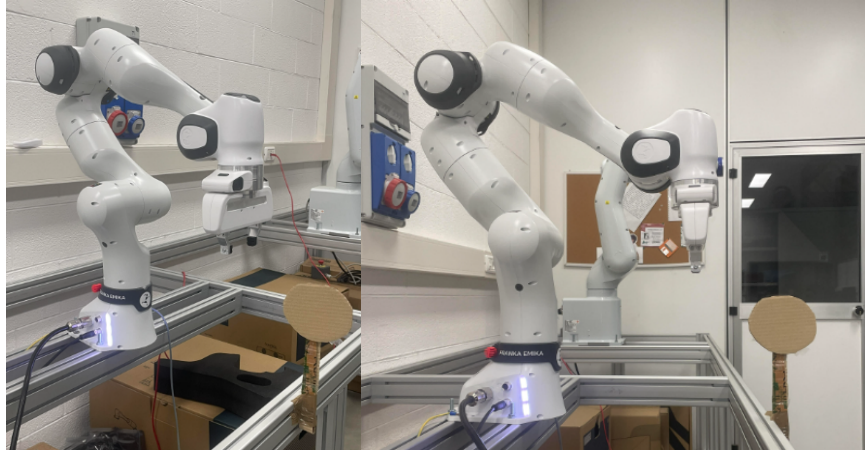


Figure 15: Initial position of the FR3 during the 2D case lab test.

The main limitations of using the real FR3 instead of the virtual environment are that the `franka_hardware` library has emergency stops when a certain acceleration and jerk are imposed onto the actuators.

These Cartesian limits are enforced by the `franka_hardware` interface and FCI firmware at the end-effector level (2 m/s^2 for acceleration and 30 m/s^3 for jerk), which dynamically projects them through the manipulator's kinematics to bound individual joint movements. Large acceleration and jerk spikes are often introduced to the velocity commands due to command discontinuity, as setpoints are passed from the 100 Hz high-level controller to the 1000 Hz low-level execution loop [13] [21].

This mismatch creates step like velocity command changes that cause the `franka_hardware` library to perform an emergency stop on the robot. This is an unavoidable safety feature of the FR3.

Therefore, extra measures to keep the acceleration within 2 m/s^2 and the jerk below 30 m/s^3 have been imposed in the lower level controller such as a ramping up of velocity commands at a lower level and also hard clamps on the acceleration and jerk that affect how the velocity commands are sent.

This is the main source of discrepancy between the results and plots ran in the virtual environment, where behaviour is unbounded and ideal, and the ones obtained by running the controller in the real robot.

Table 4: Code parameters used during the virtual testing in the 2D case using the real FR3.

Parameter	Value	Meaning/Units
dt	0.01	Control loop period [s, 100 Hz]
k_p	10.1	Task-space proportional gain
$v_{\max_task}$	0.25	Max task-space speed [m/s]
$n_{\max_dq}$	1.5	Joint velocity saturation [rad/s]
$\max_q2$	6.82	Joint 2 accel limit, incl. 1.1 safety factor [rad/s ²]
$\max_q4$	11.36	Joint 4 accel limit, incl. 1.1 safety factor [rad/s ²]
$\max_q6$	18.18	Joint 6 accel limit, incl. 1.1 safety factor [rad/s ²]
$j_{\max_q2}$	50	Joint 2 jerk limit [rad/s ³]
$j_{\max_q4}$	80	Joint 4 jerk limit [rad/s ³]
$j_{\max_q6}$	120	Joint 6 jerk limit [rad/s ³]
λ_{cbf}	7.5	CBF class-K gain (conservativeness of barrier response)
β	15	Soft-min sharpness for blending per-link barriers
$efrad_clip$	40	Max allowed norm of $\ddot{q}$ before rescaling
k_{esc}	0.15	Tangential escape velocity gain
$cstr_release$	0.2	Hysteresis release threshold for escape latch
n_{iter}	60	Bisection iterations for QP root-find
ϵ	1.00×10^{-4}	Central-difference step for barrier gradient

6.1.2. 3D experiment setup:

For this experiment setup the obstacle is kept in the same position, and the robot joints are all actuated, this means all 7 joints can move to perform the task and avoid the obstacle and the robot is free to move its links in the 3D XYZ space.

The position of the obstacle intersects with the robot manipulator's dexterous workspace, which means it's a position where the robot's EE could be placed in a multitude of poses.

By placing it in this position it is sure that the obstacle obstructs potential trajectories of the manipulator.

6.1.2.1. RVIZ 3D experiments setup:

For this first virtual experiment the `/tf` and `/joint_states` topics are published for RVIZ to read them, this way the FR3 base is placed onto the (0,0,0) coordinate or the virtual space. Then the obstacle is also published and placed onto coordinates (0.45,0,0.25).

The robot is then ordered to move its EE from cartesian position (0.30, -0.20, 0.25) to position (0.50, 0.20, 0.25) forcing thus a 3D trajectory that would collide against the obstacle if not for the collision avoidance features of the controller.

For the "cost of safety" test, the obstacle is kept in the mentioned position, but the robot follows a defined trajectory from point A to B and then C to D.

In this case the trajectory is A (0.30, -0.20, 0.25) , B (0.60,0.20 ,0.25), C (0.60, -0.20,0.25) and D (0.30, 0.0, 0.25). This trajectory forces the robot geometry to intersect with the defined obstacle and forces the controller's safety features to intervene.

A note on the results of the energy cost of safety, it has been computed taking into account the payload free energy required by the robot joints in simulation, it works only to perform a comparison between cases and it should not be taken as a correct absolute value because any variation in payload, task or robot tool would change this value.

The objective of this experiment is to observe the behaviour and performance of the controller when having to perform a task while also avoiding an obstacle. Also, to collect data on the time elapsed from the beginning to the end of the task and the energy spent to evaluate the "cost of safety".

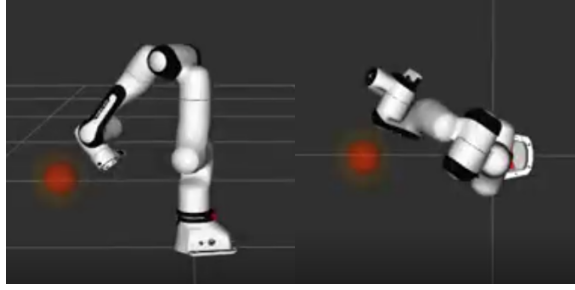


Figure 16: RVIZ representation of the FR3 in the 3D case in the starting position..

Table 5: Code parameters used during the 3D RVIZ experiment

Parameter 3D RVIZ	Value	Meaning / Units
dt	0.01	Control loop period t_d [s, 100 Hz]
K_p	1	Task-space proportional gain
$v_{\max, \text{task}}$	0.25	Max task-space speed cap [m/s]
sf	11	Safety factor divisor applied to all actuator limits
k_{post}	1	Null-space posture gain (push redundant DOF toward q_{rest})
$v_{\max, i1}$	2.382	Joint 1 velocity limit, [sf rad/s]
$v_{\max, i2}$	2.382	Joint 2 velocity limit, [sf rad/s]
$v_{\max, i3}$	2.382	Joint 3 velocity limit, [sf rad/s]
$v_{\max, i4}$	2.382	Joint 4 velocity limit, [sf rad/s]
$v_{\max, i5}$	4.782	Joint 5 velocity limit, [sf rad/s]
$v_{\max, i6}$	3.8	Joint 6 velocity limit, [sf rad/s]
$v_{\max, i7}$	4.782	Joint 7 velocity limit, [sf rad/s]
$\ddot{q}_{\max, i1}$	13.636	Joint 1 acceleration limit, [sf rad/s ²]
$\ddot{q}_{\max, i2}$	6.818	Joint 2 acceleration limit, [sf rad/s ²]
$\ddot{q}_{\max, i3}$	9.091	Joint 3 acceleration limit, [sf rad/s ²]
$\ddot{q}_{\max, i4}$	11.364	Joint 4 acceleration limit, [sf rad/s ²]
$\ddot{q}_{\max, i5}$	13.636	Joint 5 acceleration limit, [sf rad/s ²]
$\ddot{q}_{\max, i6}$	18.182	Joint 6 acceleration limit, [sf rad/s ²]
$\ddot{q}_{\max, i7}$	18.182	Joint 7 acceleration limit, [sf rad/s ²]
$\dddot{q}_{\max, i1}$	6818.2	Joint 1 jerk limit, [sf rad/s ³] – no enforcement via actuator box
$\dddot{q}_{\max, i2}$	3409.1	Joint 2 jerk limit, [sf rad/s ³] – no enforcement via actuator box
$\dddot{q}_{\max, i3}$	4545.5	Joint 3 jerk limit, [sf rad/s ³] – no enforcement via actuator box
$\dddot{q}_{\max, i4}$	5681.8	Joint 4 jerk limit, [sf rad/s ³] – no enforcement via actuator box
$\dddot{q}_{\max, i5}$	6818.2	Joint 5 jerk limit, [sf rad/s ³] – no enforcement via actuator box
$\dddot{q}_{\max, i6}$	9090.9	Joint 6 jerk limit, [sf rad/s ³] – no enforcement via actuator box
$\dddot{q}_{\max, i7}$	9090.9	Joint 7 jerk limit, [sf rad/s ³] – no enforcement via actuator box
λ_{dcbf}	15	CBF class- $\mathcal{K}$ gain (lower barrier)
β	5	Soft-min sharpness for barrier-pink linking
grad_clip	40	Max allowed norm of grad h
k_{circ}	0.2	Circulation (swirl) speed scale
h_{act}	0.1	Barrier-proximity gate width for circulation onset [rad ²]
e_0	0.06	Goal-distance fade scale; kills orbital
w_{cap}	3	Cap on the barrier-proximity gate $b(p) < h_{\text{act}}$
n_{iter}	60	Bisection iterations for CBF root-finding
ϵ	1.0×10^{-4}	Central-difference step for barrier gradient

6.1.2.2. Franka Emika FR3 lab 3D experiment setup:

For the lab experiment with the FR3 robot the position of its base is considered as the $(0,0,0)$ coordinate. The obstacle is placed on the same coordinates as in the previous experiment using the virtual environment. This way the results can be directly compared.

The robot is then ordered to move its EE from the cartesian coordinates $(0.30, -0.20, 0.25)$ to $(0.50, 0.20, 0.25)$ forcing thus a trajectory that would collide against the obstacle if not for the collision avoidance features of the controller. The obstacle is kept from previous experiments.



Figure 17: Initial position of the FR3 during the 3D case lab tests.

Each test is repeated twice placing the disk in different orientations, once in the XZ plane and once in the YZ plane to cover all the space of the theoretical 3D sphere. This is meant as a demonstration of how it avoids a 3D obstacle in physical space but practically yields the same result. For this reason, and for convenience, the result is not shown twice as it would be redundant.

Keeping in mind the hardware limitations regarding maximum acceleration and jerk the controller has been equipped with the proper measures as to avoid triggering the safety features and stopping the robot during its tasks [21].

Table 6: Code parameters used during the real testing in the 3D case using the FR3.

Parameter	Value	Meaning / Units
dT	0.01	Control loop period: 0.01 s
$K_{p,\text{task}}$	1.0	Task-space position gain (linear)
$v_{\text{max},\text{task}}$	0.25	Max task-space speed [m/s]
k_p	1.1	Saturated joint velocity limits
$q_{\text{spd},1}$	1	Posture speed limit [rad/s]
$q_{\text{rest},1}$	0.75	Posture target joint 1: q_1 [rad]
$q_{\text{rest},2}$	-0.75	Posture target joint 2: q_2 [rad]
$q_{\text{rest},3}$	0	Posture target joint 3: q_3 [rad]
$q_{\text{rest},4}$	-0.236	Posture target joint 4: q_4 [rad]
$q_{\text{rest},5}$	0.236	Posture target joint 5: q_5 [rad]
$q_{\text{rest},6}$	1.571	Posture target joint 6: q_6 [rad]
$q_{\text{rest},7}$	0.785	Posture target joint 7: q_7 [rad]
$v_{\text{joint},1}$	2.382	Joint 1 velocity limit [rad/s]
$v_{\text{joint},2}$	2.382	Joint 2 velocity limit [rad/s]
$v_{\text{joint},3}$	2.382	Joint 3 velocity limit [rad/s]
$v_{\text{joint},4}$	2.382	Joint 4 velocity limit [rad/s]
$v_{\text{joint},5}$	2.382	Joint 5 velocity limit [rad/s]
$v_{\text{joint},6}$	2.382	Joint 6 velocity limit [rad/s]
$v_{\text{joint},7}$	4.782	Joint 7 velocity limit [rad/s]
$\ddot{q}_{\text{max},1}$	1818	Joint 1 acceleration limit [rad/s ²]
$\ddot{q}_{\text{max},2}$	1818	Joint 2 acceleration limit [rad/s ²]
$\ddot{q}_{\text{max},3}$	1818	Joint 3 acceleration limit [rad/s ²]
$\ddot{q}_{\text{max},4}$	1818	Joint 4 acceleration limit [rad/s ²]
$\ddot{q}_{\text{max},5}$	1818	Joint 5 acceleration limit [rad/s ²]
$\ddot{q}_{\text{max},6}$	1818	Joint 6 acceleration limit [rad/s ²]
$\ddot{q}_{\text{max},7}$	1818	Joint 7 acceleration limit [rad/s ²]
$i_{\text{max},1}$	36.35	Joint 1 torque/current limit
$i_{\text{max},2}$	36.35	Joint 2 torque/current limit
$i_{\text{max},3}$	36.35	Joint 3 torque/current limit
$i_{\text{max},4}$	36.35	Joint 4 torque/current limit
$i_{\text{max},5}$	36.35	Joint 5 torque/current limit
$i_{\text{max},6}$	36.35	Joint 6 torque/current limit
$i_{\text{max},7}$	36.35	Joint 7 torque/current limit
λ_{cbf}	15	CBF look-ahead gain (lower = brakes earlier, higher = quicker)
β	1.0×10^{-6}	CBF-related numerical parameter

6.2. Results:

Before completing these tests, it is important to consider what are the important metrics and parameters to analyse to assess the performance of the controller, in this thesis the following have been considered:

- $h(t)$, barrier value throughout the task: this is a clear indicator of how far from the forbidden configurations the joints are. By staying above $h(t) = 0$ we can verify that none of the links or joints of the robot collide with the forbidden area (which includes the obstacle plus the safety radius) [1].
- **Correction norm**, this indicates the amount of correction, in rad/s, is imposed onto the velocity commands by the collision avoidance features of the controller, this correction activates near the boundary of the barrier function value, as the robot approaches a forbidden configuration this correction increases to keep it out of said configuration and with $h(t)$ above 0 [1].
- **Circulation or escape latch**, in m/s, this regards the activation of the local minima escape mechanism, when the robot approaches a local minimum [7].
- **EE position on the XZ and XYZ plane**, to measure how well the controller tracks the desired EE position and the route the EE takes during the task to avoid the obstacle.
- **Joint positions, velocities and accelerations** in rad, rad/s and rad/s². This is used to observe how the joints react to different tasks, gains and boundaries. These metrics are plotted during the 3D tests as in this case it is interesting to see the behaviour of the whole robot instead of a manipulator bounded in the XZ plane.

Also, this is the actual behaviour of the robot under this controller since the 2D version signifies a stepping stone towards the complete 7 joints controller.

Finally, these results are compared with the ones obtained using a nominal P position controller actuated with velocity commands. The objective is to compare the performance and have a baseline of how a normal controller would act in presence of the obstacle.

6.2.1. 2D experiment results

For this experiment it is shown that $h(t)$ stays above 0, thus certifying the safety of the controller avoiding the obstacle. To do this, the correction imposed by the controller is also plotted to see the magnitude of the correction and when does it activate. The tangential escape latch is also plotted to observe the escape mechanism turning on.

Further verification is conducted plotting the EE trajectory and final position tracking to assess performance and desired behaviour.

The four most relevant graphs are the following:

- **EE trajectory**; it draws the trajectory the end-effector follows to complete the task in the workspace with an obstacle present.
- **$h(t)$** , barrier value throughout the task
- **Correction norm**, in rad/s, the amount of velocity added onto the final velocity signal sent to the lower-level controller by effect of the CBF safety features. It illustrates how much the CBF affects the nominal velocity.
- **Escape latches** engage.

6.2.1.1. RVIZ 2D experiment results:

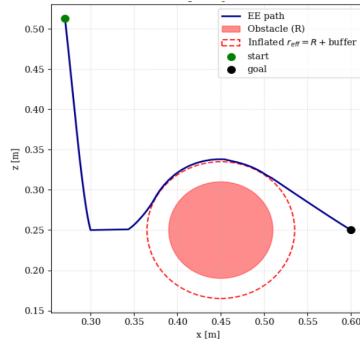


Figure 18: Trajectory of the EE during the task where it avoids the obstacle area.

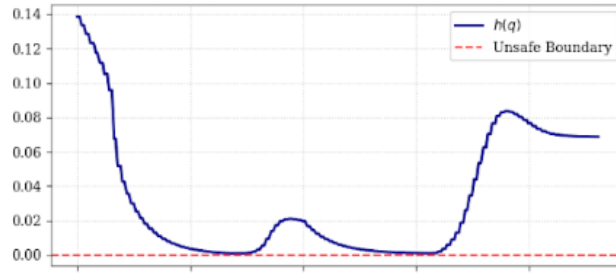


Figure 19: Evolution of the CBF barrier $h(t)$ during the task, where $h(t)$ stays above 0, signalling safety barrier compliance

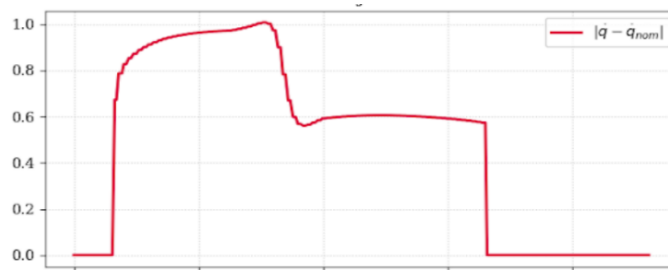


Figure 20: Correction applied to the nominal velocity in rad/s.

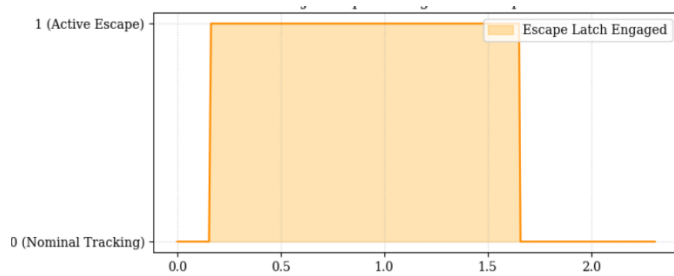


Figure 21: Engagement of the escape latch during the task. If 0 – not active; if 1 – active.

In this first experiment we see how during the task portion where the robot would be forced to collide against the forbidden area, it reactively avoids it thanks to the CBF safety barrier. It does so by implementing a tangential component of 0.0 to 1.0 rad/s (depending on proximity to the safety barrier) to the nominal velocity that causes the robot geometry to flow alongside the obstacle barrier, keeping thus the $h(t)$ above 0 for all the task. Additionally the escape latch activates throughout most of the task to avoid a local minima.

6.2.1.2. Franka Emika FR3 2D lab results

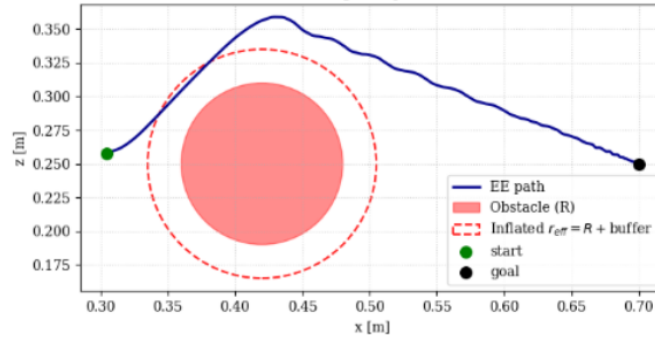


Figure 22: Trajectory of the EE during the task where it avoids the forbidden area during the task.

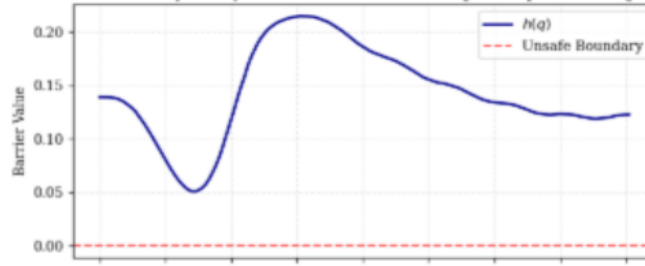


Figure 23: Evolution of the CBF barrier $h(t)$ during the task where $h(t)$ stays above 0.

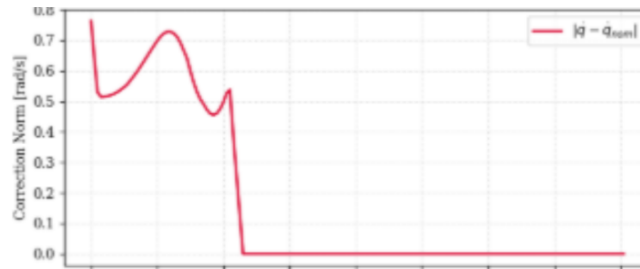


Figure 24: Correction applied to the nominal velocity in rad/s.

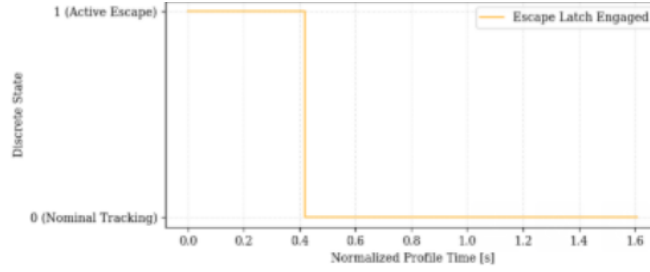


Figure 25: Engagement of the escape latch during the task. If 0 – not active; if 1 – active..

In this experiment, the task is equivalent to that of the previous RVIZ experiment but recreated, with caveats, using the real Franka Emika FR3 in the Intelligent Robotics lab at Universita di Pavia. As seen in the parameters, the physical robot has some limitations that the RVIZ virtual one does not, velocity, acceleration and jerk limits that can be seen in Table 4. This causes us to have to act more conservatively to avoid the obstacle since the robot will not be able to react as aggressively. For this reason a lambda of 7.5 instead of the initial 15 has been imposed.

This has allowed the robot geometry to fully avoid the obstacle area and $h(t)$ to remain above 0, thanks to the safety barrier. This has been accomplished by the controller by adding a velocity component of 0.0 to 0.7 rad/s (depending on proximity to the safety barrier) to the nominal velocity required to complete the task. Furthermore, in this case the escape latch activates briefly in the most critical moment of the task where it could have created a local minima.

The difference in trajectory compared to the RVIZ experiment is given by the difference in lambda and the robot servomotor physical limitations.

6.2.1.3. Comparison against a nominal P controller

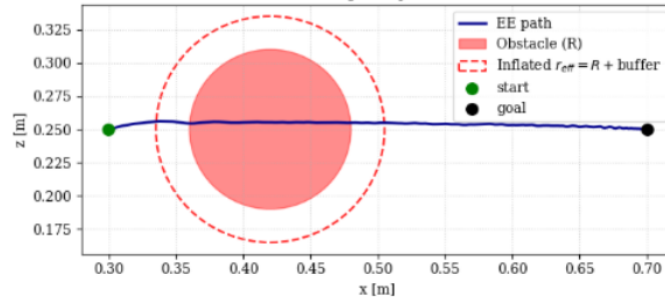


Figure 26: Trajectory of the EE during the task where it avoids the forbidden area during the task.

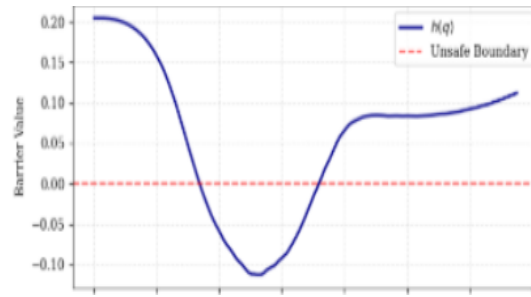


Figure 27: Evolution of the CBF barrier $h(t)$ during the task where it crosses below 0.



Figure 28: Null correction applied to the nominal velocity in rad/s.

This experiment shows the behaviour of the robot arm when no safety barrier is imposed. It is used as a practical comparison to observe the impact of the CBF.

It is observed that the robot's geometry does not avoid the obstacle and instead it collides against it fully and how the safety barrier is violated. This can also be observed because $h(t)$ goes below zero which is the condition that regulates the safety condition.

Also, the correction applied is 0 rad/s since the barrier is not active so it never applies any correction to the nominal velocity imposed by the controller in order to complete the task.

6.2.2. 3D experiment results

For this experiment it is shown that $h(t)$ stays above 0 also when the seven joints are actuated and the robot is free to move in 3D space, thus certifying the safety of the controller avoiding the obstacle. To do this, the correction imposed by the controller is also plotted to see the magnitude of the correction and when does it activate. The circulation method is also plotted to observe the escape mechanism turning on.

Further verification is conducted plotting the EE trajectory and final position tracking to assess performance and desired behaviour.

The six most relevant graphs are the following:

- **EE trajectory**; it draws the trajectory the end-effector follows to complete the task in the workspace with an obstacle present.
- **$h(t)$** , barrier value throughout the task
- **correction norm**, in rad/s, the amount of velocity added onto the final velocity signal sent to the lower-level controller by effect of the CBF safety features. It illustrates how much the CBF affects the nominal velocity.
- **Escape latches engage**.
- **Time duration and completion** % of the task for the RVIZ case.
- **Energy spent** during the task for the RVIZ case.

6.2.2.1. RVIZ 3D experiment results

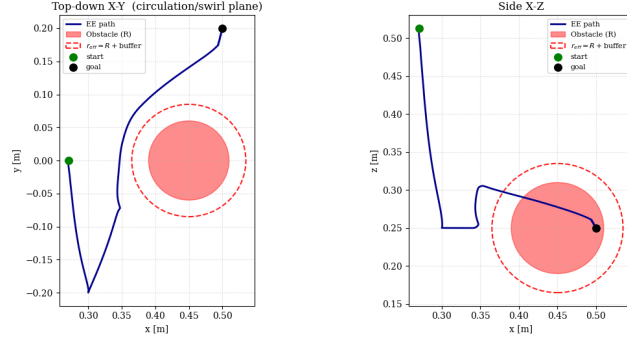


Figure 29: Trajectory of the EE in 3D during the task where it avoids the forbidden area during the task.

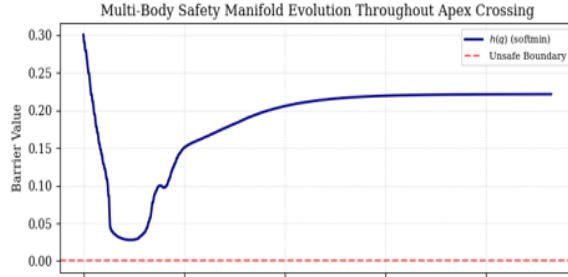


Figure 30: Evolution of the CBF barrier $h(t)$ during the task where it does not cross below 0.

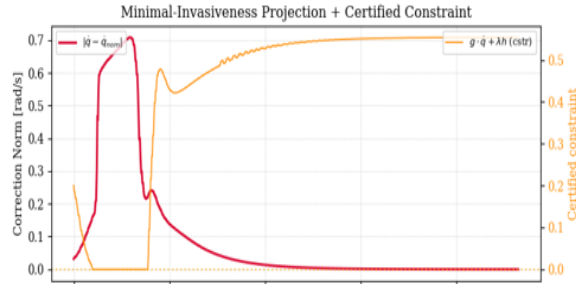


Figure 31: Correction applied to the nominal velocity in rad/s and certified constraint.

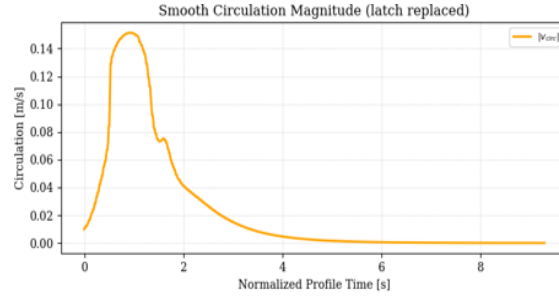


Figure 32: Smooth circulation magnitude in 3D.

In this experiment the task is expanded to a 3D space. Here it is shown that it avoids the obstacle area also when travelling in the XYZ plane and the safety barrier ($h(t)$) stays above zero, showing thus safety compliance.

It does so by adding a velocity component of 0.0 to 0.7 rad/s to the nominal velocity imposed by the controller to fulfill the task. Furthermore in this case the 2D escape latch is exchanged for a smooth circulation mechanism more apt for the 3D case and it is shown that it activates towards the most critical configuration to avoid creating a local minima and getting stuck.

6.2.2.2. RVIZ 3D cost of safety results

For this case the (A,B,C,D) trajectory is completed with the custom controller and with a nominal velocity controller. The results in terms of time and energy spent are compared to see how much it would cost to add this feature in a continuous operation industrial setting.

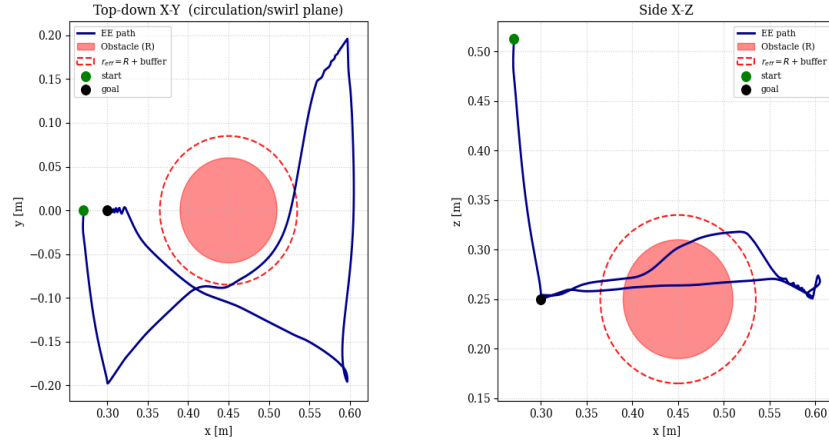


Figure 33: Trajectory of the EE in 3D during the ABCD task where it avoids the forbidden area.

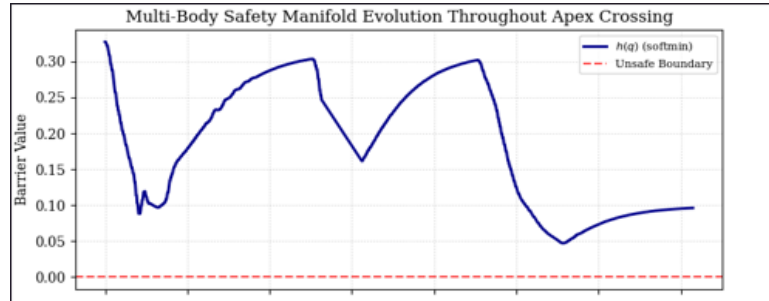


Figure 34: Evolution of the CBF barrier $h(t)$ during the ABCD task where it stays above 0.

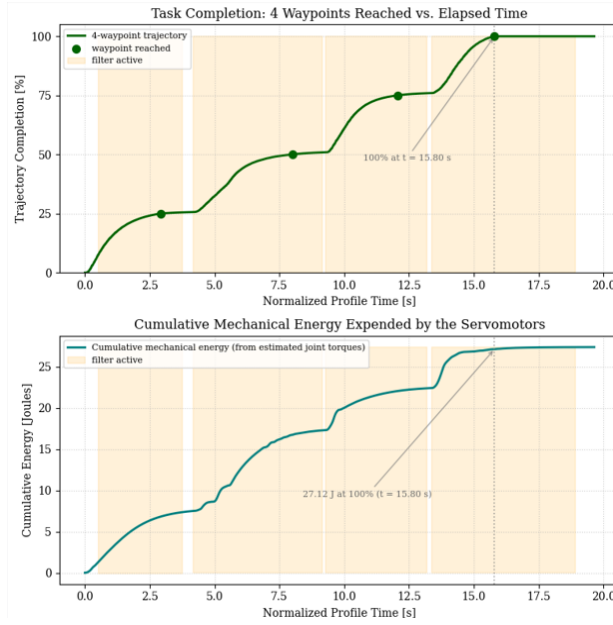


Figure 35: Task completion percentage over time and energy spent during the ABCD task.

During this experiment not only safety compliance is evaluated but also how much energy and time it takes to complete the task while the safety barrier is active. Evaluating these two extra parameters is crucial since are some of the most important during continuous operation in an industrial setting and allow us to consider the "cost of safety".

It can be observed that the safety compliance is kept, as expected since it is the same setting, from the previous experiments and that the robot geometry avoids the forbidden area during the task. Also it is shown that $h(t)$ never goes below zero signaling that the safety barrier is never violated during the task.

Additionally, it is shown that the task takes a total of 15.80s and consumes a quantity of 27.12J (this energy quantity is used for a comparison with a different experiment but would certainly change in different conditions and payloads).

6.2.2.3. Cost of safety of the nominal controller implemented and tested in RVIZ

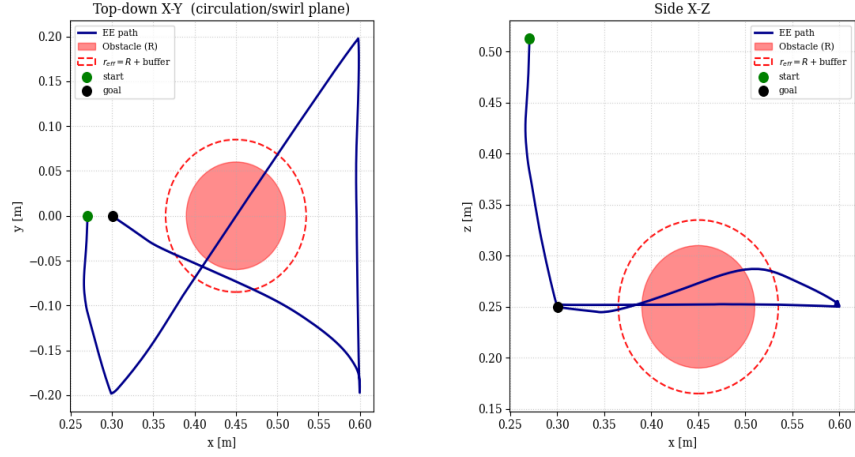


Figure 36: Trajectory of the EE in 3D during the ABCD task where it crosses the forbidden area.

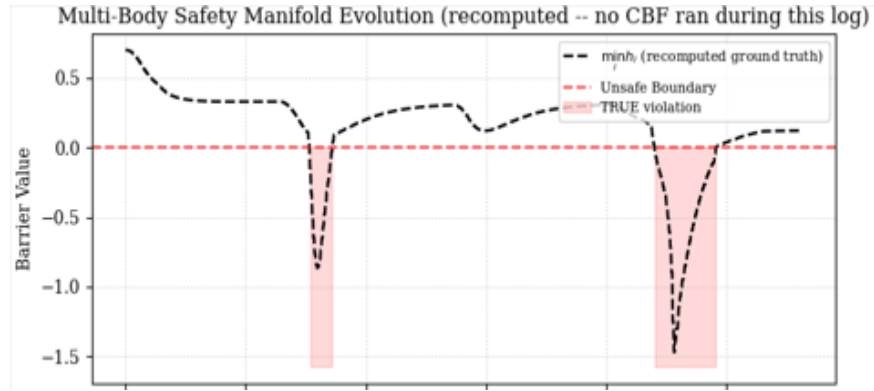


Figure 37: Evolution of the CBF barrier $h(t)$ during the ABCD where it crosses below 0.

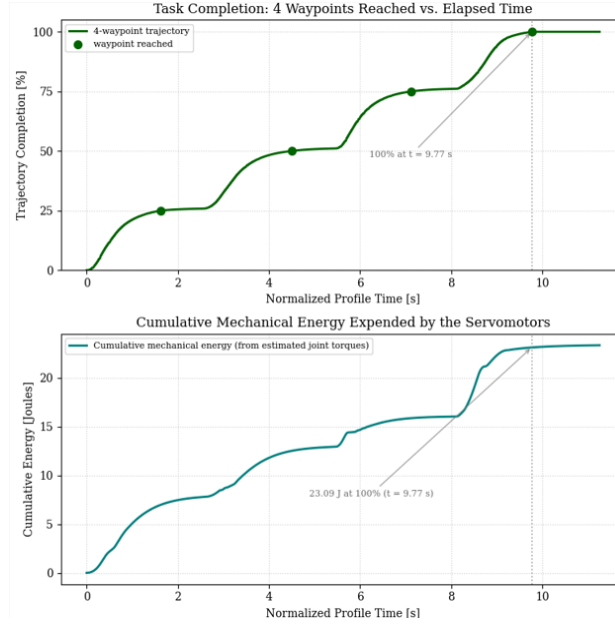


Figure 38: Task completion percentage over time and energy spent during the ABCD task.

During this experiment the same ABCD task is completed but in this case without the safety barrier. The target is to evaluate the time and energy spent when using only a nominal controller.

In this case we see the robot's geometry enters the forbidden area or "collides" and also $h(t)$ goes below 0, therefore the safety barrier is violated.

Also we can see that the task takes 9.77s to be completed and it consumes around 23.09J (this energy quantity is used for a comparison with a different experiment but would certainly change in different conditions and payloads). This shows that the controller that features the safety barrier is more "expensive" in terms of time and energy.

6.2.2.4. Franka Emika FR3 3D lab results

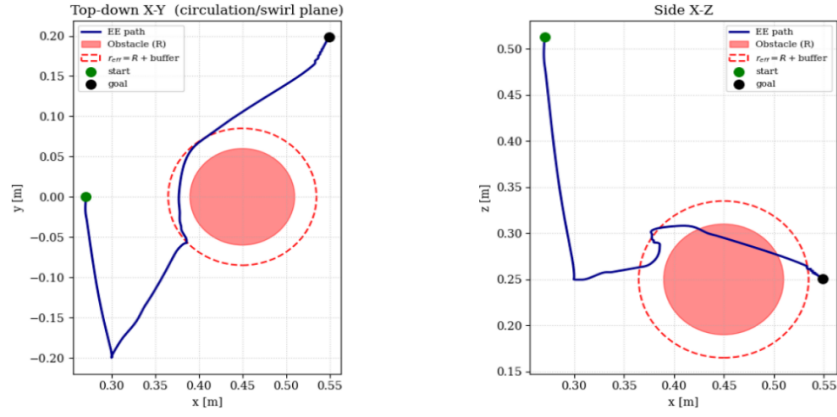


Figure 39: Trajectory of the EE in 3D during the task where it avoids the forbidden area.

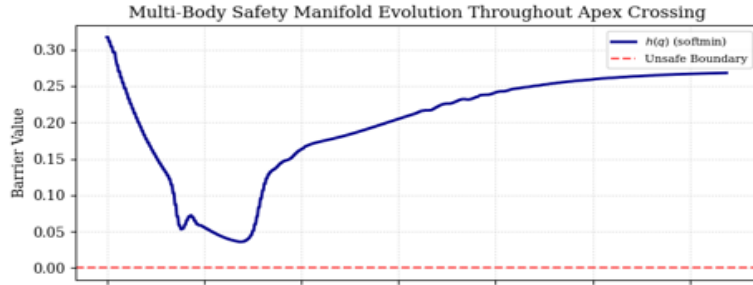


Figure 40: Evolution of the CBF barrier $h(t)$ during the task where it never crosses below 0.

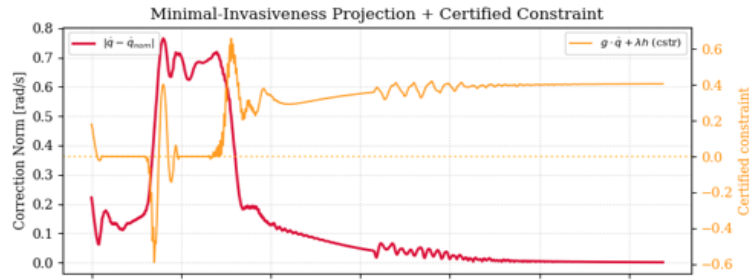


Figure 41: Correction applied to the nominal velocity in rad/s.

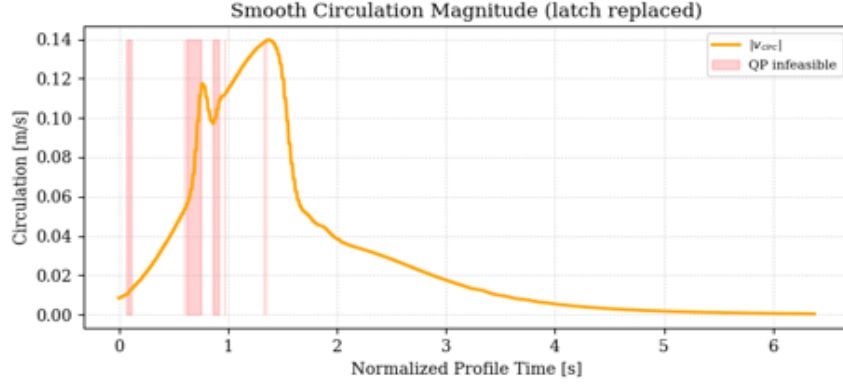


Figure 42: Smooth circulation magnitude in 3D.

In this experiment the task previously tested in RVIZ is done using the real FR3 robot arm. It is observed that the robot arm never goes in the forbidden area and that $h(t)$ stays above 0. This shows that the safety constraint is never violated during the task.

To do so a velocity component of 0.0 to 0.7 rad/s (depending on proximity to the safety barrier) is added to the nominal velocity imposed by the controller in order to complete the task.

Furthermore the smooth circulation mechanism activates progressively as the robot's geometry approaches a position where it could cause a local minima thus blocking the robot in position.

In this case, the λ is kept at 15 as in the RVIZ case as the larger mobility capabilities in the 7-DoF arm causes the robot arm to be able to react more aggressively and avoid the obstacle area without saturating its actuators by going beyond the velocity, acceleration and jerk limits. For this reason the results look much more similar between the virtual and physical robot than in the 2D case, the main differences being caused by noise in the feedback and the controller's reaction to said noise.

6.2.2.5. Comparison against a nominal controller

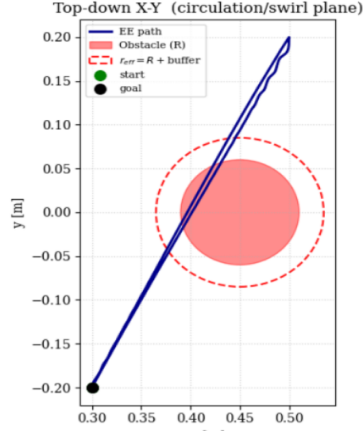


Figure 43: Trajectory of the EE in 3D during the task where it crosses the forbidden area.

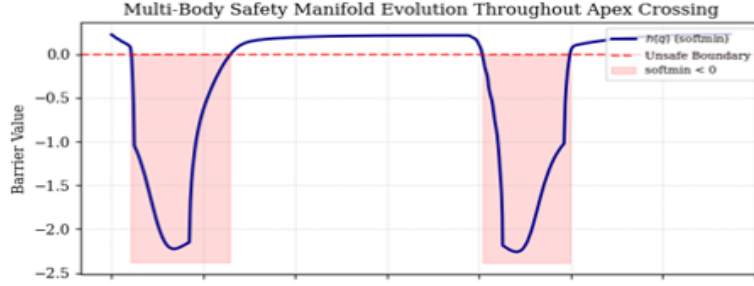


Figure 44: Evolution of the CBF barrier $h(t)$ during the task where it crosses below 0.

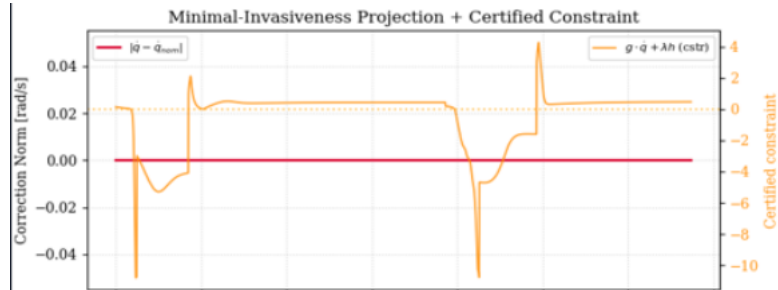


Figure 45: Null correction applied to the nominal velocity in rad/s.

During this experiment the nominal controller without the safety barrier is tested. Here it shows that it crosses or "collides" into the forbidden area and also $h(t)$ crosses below zero, this means it violates the safety barrier.

Furthermore, the correction stays at 0.0 rad/s during the task because if the barrier is not active the correction never activates.

6.3. Engineering findings and hardware limitations:

This chapter aims to discuss the most important engineering findings from the extensive testing of the controller and the limitations found when performing the tasks on the real FR3 robot arm.

The main hardware limitations that the real Franka Emika FR3 presents are the velocity, acceleration and jerk constraints. In the ideal case, a RVIZ simulation of a robot with the same geometry as the FR3, one could order the robot to perform a movement or react to a constraint in an ideal way, imposing high velocities to the robot joints, creating thus high acceleration and jerk values.

This is not possible in the real world since the robot has physical limitations and furthermore, the *franka_hardware* library has very strict constraints (2 m/s² acceleration, 30 m/s³ jerk) to keep the robot moving well within its hardware limitations. These limits are imposed at each joint in rad/s² and rad/s³, but its values depend on the kinematics of the robot.

This means that if at any instant of time during the task the high level CBF constrained controller orders the lower-level velocity controller to impose an “out-of-bounds” velocity to fulfil its safety requirements, the robot arm gets instantly blocked and the task is interrupted. It does not allow the actuators to saturate but instead blocks all joints.

This forces the controller developer to include velocity limits to the high-level controller as well so as to not generate any forbidden motion that would potentially block the robot. This is the main source of divergence between the results with the tests performed in the RVIZ environment and the tests performed with the real FR3.

Other mentionable sources of this divergence are sensor noise that affect the feedback the controller gets and the non-ideal behaviour of the electrical drives and actuators

6.4. Failure modes and research hurdles log:

The failure modes of this controller encountered during the development and testing of the controller are as follows:

- **Velocity, acceleration and jerk constraints not respected**, and robot gets blocked (this has been addressed and solved constraining the maximum values of these parameters on the controller level).
- **Lambda too large when dynamics are constrained**. This is a real limitation of the robot and must be properly tuned. Being that the robot is not able to perform at its full capacity because of the dynamic hardware constraints, the lambda value has to be tuned to allow the robot to react with enough time since it cannot react as aggressively as the controller would suggest respecting the safety barrier.
- **Vibration, instability and overshoot** issues when Kp too large, this is a noted P controller issue that must be properly addressed to tune Kp correctly, in the controller the Kp is set at values from 10.0 to 1.0 depending on the controller task and features. This creates a controller with a slower response but a stable performance that is critical to the safety features. This is an intentional design decision since the critical feature of the controller at this stage is safety and not speed.
- **Discontinuity reflex block**, since the higher level controller passes information at 100Hz and the low level velocity controller passes its velocity signal at 1000Hz to the robot joints (this is ROS2 control standard design and cannot be modified), if the values of velocity change too abruptly from tick to tick on the high level controller it creates high acceleration and jerk values on the lower level controller and the franka_ hardware library blocks the joints using its emergency safety features. To address this, a gradual ramp up of the velocity, acceleration and jerk values as well as constraints have been added to the lower level controller to make the ramp up smooth in the 10 ticks between the 100Hz higher level controller and 1000Hz lower-level controller instead of an abrupt step caused by this discontinuity.

6.5. Results discussion and conclusions:

After performing the programmed tests to assess performance, the following can be interpreted from the results.

The controller is properly constrained by the CBF safety features and by keeping $h(t)$, defined as a radial distance between the current angle and the forbidden configuration angle “cone”, above 0, the controller is able to keep the robot from colliding with an obstacle defined in 3D space.

Also, the user can configure how conservative the controller is by modifying λ to make it pass nearer or further from the obstacle. From the results it can be observed that a conservative λ is necessary for the real hardware case since velocity and acceleration constraints impedes the controller reacting ideally so keeping a larger buffer by having λ small is necessary to respect the safety boundary.

Other parameters and gains such as K_p and maximum velocity, jerk and acceleration of the controller can also be imposed on the controller, and it will adapt its behaviour to the required specifications keeping the safety feature intact and behaving always in the defined “safe way”. Overall, the controller can be considered robust and safe.

The escape latch and circulation are effective methods to solve the local minima issue, as seen in the testing when the robot arm is approaching a local minimum, both methods activate to “pull” it out of that situation and allow it to continue with the task.

When **compared to a nominal controller** it can be observed that the CBF creates an extra velocity component that corrects the trajectory and keeps the robot from colliding against the defined obstacle. The nominal controller does not generate this extra velocity and thus it follows a straight trajectory that eventually collides with the obstacle.

Goal tracking is accurate as long as it does not interfere with the safety constraints. The final position of the end-effector is accurate to the order of 0.5mm for all tests unless the final position commanded is unreachable while respecting the safety constraints. In the cases where the commanded position is unreachable, the end-effector stays at the nearest possible position that does not violate the safety barrier.

This result is very relevant as it illustrates the hierarchy in the controller’s priority, safety first, tracking second. This is the desired hierarchy when designing a safety critical controller that must interact with humans in the same workspace.

The cost of safety in terms of time and energy are considerable (+61.7% for the time and +17.8% for the energy costs), and it has been computed for the A,B,C,D task, so it can be said that this controller is safe for collision avoidance of a set obstacle but also more expensive to run.

Metric	Baseline (no filter)	CBF (filtered)	Cost
Time	9.77 s	15.8 s	+61.7%
Energy	23.02 J	27.12 J	+17.8%

Table 7: Comparison of time and energy consumption with and without CBF filtering.

Chapter 7 – Conclusions and Future Work

7.1. Summary of contributions

This thesis presented the design, implementation, and experimental validation of a real-time, multi-rate safety control framework for 3D reactive obstacle avoidance on the 7-DoF Franka Emika FR3 manipulator [21]. The primary objective was to establish rigorous collision avoidance guarantees across the full physical volume of the arm while preserving high-level task tracking and resolving spurious local minima deadlocks [1].

The main technical contributions of this work are summarized as follows:

- **3D Angular Barrier Formulation with Softmin Aggregation:** Extended Capisani angular barrier metrics to non-point-mass link geometries by modelling sequential arm links as spatial cylinders using $h_i(q) = \phi_i^2 - \text{sgn}(\delta_i) \delta_i^2$ (47). Individual link barriers were unified into a single differentiable constraint using whole-arm soft-minimum log-sum-exp aggregation $h(q)$ (50), eliminating non-smooth switching behaviour during multi-obstacle or multi-link avoidance manoeuvres.
- **Deadlock Resolution via Goal-Attenuated Circulation Fields:** Synthesized a continuous, goal-attenuated spatial tangential circulation field v_{circ} (60) using normalized swirl vector t injected directly into the nominal task-space reference velocity (61) prior to differential IK resolution (3). This mechanism systematically breaks collinear deadlock alignments ($\dot{q}_{nom} \parallel -\nabla h(q)$) near obstacle boundaries, allowing the end-effector to bypass topological saddle points and local minima without chattering or goal orbiting.
- **Multi-Rate Asynchronous Software Architecture:** Implemented a decoupled control stack separating high-level safety filtering from low-level motor execution. The 100 Hz Python Application Layer leverages Pinocchio rigid-body dynamics and a scalar-dual Quadratic Program (QP) filter to compute certified velocity setpoints, while the 1000 Hz C++ Hardware

Interface Layer (ros2_control and libfranka) enforces deterministic jerk-limited execution over the Franka Control Interface (FCI) [28].

- **Experimental Verification and Diagnostic Pipeline:** Developed an automated offline diagnostic pipeline (plot_results.py) to extract telemetry, evaluate strict forward invariance, quantify temporal and energy overheads, and log automated execution gap verdicts.

7.2. Key Findings and System performance

The operational performance of the control architecture was evaluated across multiple workspace trajectories and velocity profiles. Telemetry analysis yielded several key findings:

- **Forward Invariance of the Safe Set:** The scalar-dual QP safety filter successfully maintained $h_{\text{soft}}(q) \geq 0$ across all nominal workspace trajectories. The safety layer dynamically modified nominal task velocities only when approaching barrier boundaries, ensuring unperturbed trajectory execution throughout unconstrained regions [1].
- **End-Effector Target and Pose Convergence:** Across all test scenarios, the framework demonstrated reliable Cartesian convergence of the end-effector position p_{ee} toward the desired target pose p_{goal} . The goal-attenuation gate smoothly decayed the tangential swirl velocity to zero as tracking error diminished ($\|e\| \rightarrow 0$), preventing goal orbiting, sign-test chattering, or residual steady-state offset once the obstacle boundary was cleared [2].
- **Computational Efficiency:** The QP filtering problem is reduced to a single-variable dual search ($\mu^* \geq 0$). This maintained a substantial computational margin beneath the 10 ms execution deadline of the 100 Hz application loop [28].
- **Smooth Local Minima Escape:** Incorporating dual-gated circulation fields eliminated local minima trapping. The obstacle proximity gate activated swirl velocity exclusively within the boundary layer, while the goal attenuation gate smoothly deactivated circulation as the end-effector converged to p_{goal} , achieving exact asymptotic convergence [9] [27].
- **Actuator Bound Adherence:** Dynamic velocity bounding successfully mapped joint position, velocity, acceleration, and jerk limits into an instantaneous velocity interval. Low-pass exponential filtering prevented command step-discontinuities, resulting in smooth motor velocity commands and chatter-free hardware execution [21].

7.3. Limitations and critical assessment

Despite successful real-time performance, several theoretical and practical limitations were identified during system deployment:

- **Geometric Conservatism of Cylinder Bounds:** Representing the complex spatial envelope of the Franka FR3 using five cylindrical link primitives introduces geometric conservatism. While this guarantees spatial safety, it artificially restricts the robot's allowable configuration space near tight obstacles.
- **Finite-Difference Gradient Approximations:** Evaluating the soft-min barrier gradient $\nabla_q h_{\text{soft}}(q)$ via central finite differences introduces numerical sensitivity near anti-parallel vector alignments ($\theta_i \rightarrow \pi$). While gradient norm clipping stabilized velocity setpoints, an analytical Jacobian formulation would improve numerical robustness and reduce computation cycles.
- **Static and Known Obstacle Assumption:** The current safety filter assumes obstacle positions (V) and effective radii (R_{obs}) are known *a priori* and fixed in world coordinates. The architecture does not currently account for dynamic obstacle trajectories or sensor noise originating from online perception pipelines.
- **Low-Level Actuator Lag:** Telemetry revealed occasional minor lag between published 100 Hz velocity setpoints and physical joint actuation under high acceleration regimes. While firmware-level limits prevented damage, actuator lag can temporarily diminish certified constraint margins if execution delays accumulate.

7.4. Future Work and Research Directions

While the current safety filter works well on the physical robot, several practical improvements can be made in future iterations:

- **Adding 3D Cameras (Vision Integration):** Currently, obstacle positions and sizes must be known in advance. The most important next step is mounting 3D cameras (like RGB-D depth sensors) on or around the robot [29]. The live camera data can be turned into 3D point clouds or bounding shapes, allowing the safety filter to detect unknown or moving obstacles automatically [29, 30]. This could build the obstacle shapes to feed directly into the CBF-based controller and thus integrate the full working pipeline; observe the workspace - recognize obstacles - map the obstacles in the joint space - implement CBF-based safety filter to velocity commands - verify results and performance.
- **Adding a Global Path Planner:** Currently, the robot moves directly toward its target and relies on the safety filter to push it around obstacles. Linking the local CBF filter with a global path planner (such as A^* or RRT*) would allow the robot to choose the shortest or lowest-energy route around large obstacles, while the CBF filter acts as a final safety layer [31].
- **Handling Moving Obstacles:** The current barrier equations assume obstacles stay in one place. Updating the math to track obstacle speed and movement direction will allow the robot to predict where moving objects, or human workers, are going and react smoothly in real time [17].
- **Moving to Torque-Level Control:** The safety filter currently sends joint velocity commands to the robot at 100 Hz. Controlling motor torque directly at 1000 Hz would make the robot's movements smoother and allow it to handle external forces, soft physical contacts, and motor strength limits more safely [17].

7.5. Final Words

In conclusion, this thesis successfully bridges the gap between continuous Control Barrier Function theory and real-time execution on a physical 7-DOF manipulator. By combining 3D angular link geometry [11], smooth softmin aggregation [12, 22], and deadlock-free circulation fields [10, 19] into a multi-rate ROS 2 control architecture [28], the system proves that strict collision safety can be guaranteed without sacrificing movement efficiency or target accuracy [1, 2]. As robotic arms increasingly transition from structured factory cages into unpredictable human environments [20], real-time safety layers like the one developed in this work will serve as an essential foundation for building reliable, trustworthy, and safe robotics systems [1, 17].

Bibliography

- [1] F. Ferraguti, C. Talignani Landi, A. Singletary, H.-C. Lin, A. Ames, C. Secchi, and M. Bonfè, "Safety and efficiency in robotics: The control barrier functions approach," *IEEE Robotics & Automation Magazine*, vol. 29, no. 3, pp. 139–151, Sept. 2022.
- [2] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. London, U.K.: Springer-Verlag, 2009.
- [3] A. Ferrara, "Robot Control," Course 504463 Lecture Notes, Department of Electrical, Computer and Biomedical Engineering, University of Pavia (UNIPV), Academic Year 2025–2026.
- [4] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017.
- [5] R. Haschke, E. Weitnauer, and H. Ritter, "On-line planning of time-optimal jerk-limited trajectories," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 2555–2560.
- [6] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, Mar. 1986.
- [7] W. Wang, Y. Li, and X. Zhang, "An effective manipulator trajectory planning with obstacles using virtual potential field method," in *IEEE International Conference on Mechatronics and Automation (ICMA)*, 2007, pp. 3548–3553.
- [8] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.
- [9] M. F. Reis, A. P. Aguiar, and P. Tabuada, "Control barrier function-based quadratic programs introduce undesirable asymptotically stable equilibria," *IEEE Control Systems Letters*, vol. 5, no. 2, pp. 731–736, 2021.
- [10] V. M. Gonçalves, P. Krishnamurthy, A. Tzes, and F. Khorrami, "Control barrier functions with circulation inequalities," *IEEE Transactions on Control*

Systems Technology, vol. 32, no. 4, pp. 1426–1441, 2024.

[11] L. M. Capisani, T. Facchinetti, A. Ferrara, and A. Martinelli, "Obstacle modelling oriented to safe motion planning and control for planar rigid robot manipulators," *Journal of Intelligent & Robotic Systems*, vol. 71, no. 2, pp. 159–178, 2013.

[12] P. Rabiee and J. B. Hoagg, "Soft-minimum barrier functions for safety-critical control subject to actuation constraints," in *American Control Conference (ACC)*, 2023, pp. 4272–4277.

[13] D. Štogl, B. Maguire, B. S. A. Bote, and D. Coleman, "ros2_control: A robot-agnostic control framework for ROS 2," in *ROSCon*, 2021.

[14] Open Source Robotics Foundation (OSRF), "URDF: Unified Robot Description Format," ROS Documentation, 2023. [Online]. Available: <http://wiki.ros.org/urdf>

[15] A. Liégeois, "Automatic control of velocity and configurations of multi-body open-chain manipulators," *IEEE Transactions on Automatic Control*, vol. 22, no. 6, pp. 868–871, 1977.

[16] C. W. Wampler, "Manipulator inverse kinematic solutions based on damped least-squares and second-order Taylor series approximations," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 1, pp. 93–101, 1986.

[17] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *European Control Conference (ECC)*, 2019, pp. 3420–3431.

[18] G. Notomista and M. Egerstedt, "Persistently exploring complex environments using multi-robot systems," in *IEEE Conference on Decision and Control (CDC)*, 2018, pp. 3682–3687.

[19] S. Srinivasan, A. Dabholkar, and S. Coogan, "Control barrier functions for dynamic obstacle avoidance using circulation fields," *IEEE Control Systems Letters*, vol. 5, no. 6, pp. 1964–1969, 2021.

[20] F. Ferraguti, C. Talignani Landi, L. Sabattini, C. Secchi, and C. Fantuzzi, "Safety barrier functions for human–robot interaction with industrial manipulators," *European Journal of Control*, vol. 55, pp. 12–21, 2020.

[21] Franka Robotics GmbH, Hardware Manual Franka Research 3 (Document No. R02210), Munich, Germany, 2025.

[22] F. Calvia, A. D. Ames, and C. Belta, "Smooth Multi-Constraint Control Barrier Functions with Applications to Robot Manipulation," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1213–1218, 2020.

[23] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

- [24] S. M. Khansari-Zadeh and A. Billard, "A Dynamical System Approach to Real-Time Obstacle Avoidance," *Autonomous Robots*, vol. 32, no. 4, pp. 433–454, 2012.
- [25] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 398–410, 1997.
- [26] D. E. Koditschek and E. Rimon, "Robot Navigation Functions on Manifolds with Boundary," *Advances in Applied Mathematics*, vol. 11, no. 4, pp. 412–442, 1990.
- [27] F. A. Goodarzi and D. Lee, "Smooth vector field navigation and obstacle clearance for robotic manipulators," *Autonomous Robots*, vol. 46, no. 3, pp. 415–430, 2022.
- [28] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.
- [29] Open Source Robotics Foundation (OSRF), "ROS 2 Launch System Architecture and Specification (XML/YAML/Python)," *ROS 2 Documentation*, 2023. [Online]. Available: <https://docs.ros.org>
- [30] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.
- [31] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006.
- [32] G. Notomista and M. Egerstedt, "Constraint-driven torque control for safe human-robot interaction using control barrier functions," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 2110–2122, 2021.