Example of PhD Thesis with RoboticsLaTeX template

Università
di Genova

Simone Lombardi

DIBRIS - Department of Computer Science,
Bioengineering, Robotics and System Engineering

University of Genova

Supervisors:
Prof. Giorgio Cannata
PhD. Francesco Grella
PhD. Francesco Giovinazzo

In partial fulfillment of the requirements for the degree of

Laurea Magistrale in Robotics Engineering

December 17, 2025

Declaration of Originality

I, Simone Lombardi, hereby declare that this thesis is my own work and all sources of information and ideas have been acknowledged appropriately. This work has not been submitted for any other degree or academic qualification. I understand that any act of plagiarism, reproduction, or use of the whole or any part of this thesis without proper acknowledgment may result in severe academic penalties.

Acknowledgements

I want to thanks all the people that helped me during my time at University of Genova, starting with professor Cannata. His assistance was essential in the development of this thesis. I than extend my depest gratitude to Francesco Grella and Francesco Giovinazzo, them with all the other people of the MACLAB laboratory made me feel welcomed and have given me invaluable advice throughout my journey with them.

On a personal note, I want to thanks all my colleagues of the Robotics Engineering course. The friendship I found are extremely meaningful to shape me in the person I am today. Last but not least, in the slightest I want to tell my family and friends that their unwavering support and belief in me did not go unnoticed, I would not be here today if it wasn't for them.

This is a short, optional, dedication. To all the Master and PhD students of Robotics Engineering at the University of Genova.

Abstract

Since the 1960s, the use of robotic systems in industrial applications has continuously increased. However, even with this incredible force driving innovation, some tasks have proven to be too complex or not cost-effective to be performed by a robot. With the advent of Industry 4.0, the proposed solution to these problems was **Human-Robot Collaboration** — building work-cells capable of integrating a human agent performing a set of tasks that can be coordinated with a robotic agent to achieve a common objective. This approach opened up a completely new set of challenges, the first of which are safety and perception. The robotic agent needs a way to perceive the human in the workcell and must be able to react to unpredictable movements to avoid collisions. During my thesis, I worked within the **SESTOSENSO project**, specifically in Use Case 1. Their robotic system, composed of two 6-DoF industrial articulated robots mounted in series, is equipped with a set of proximity and tactile sensors. My work focused on creating a unified architecture for the two robots, exploring the capabilities of a 12-DoF robot, and proposing possible directions to improve the system's functionalities. Moreover, this work also aimed to identify potential problems and weaknesses. I achieved these objectives through a series of simulated experiments, using a task-priority approach for system control, as I was interested in exploiting the high redundancy of the robot to perform multiple tasks simultaneously. I than analyzed the result to evaluate the effect of each task on the behavior of the robot.

Contents

1 Introduction **1**
- 1.1 Research problem and objective . . . 2
- 1.2 Thesis structure . . . 2

2 State of the art **3**
- 2.1 Environment perception and awareness . . . 3
  - 2.1.1 Image recognition based methods . . . 3
  - 2.1.2 Point-cloud discretization based methods . . . 6
- 2.2 Obstacle avoidance in HRC . . . 6
  - 2.2.1 Redundancy control . . . 8
- 2.3 High DoF architecture . . . 8
  - 2.3.1 Dual-arm systems . . . 9
  - 2.3.2 Snake-like robot . . . 10
  - 2.3.3 Planar robot . . . 10
  - 2.3.4 Macro Micro configuration . . . 12

3 Architecture implementation **15**
- 3.1 System Description . . . 15
- 3.2 JointRobotTP Class Implementation . . . 17
  - 3.2.1 Class setup . . . 17
  - 3.2.2 Data structure . . . 18
  - 3.2.3 Task state update . . . 18
- 3.3 Action server . . . 19
  - 3.3.1 Action Server process flow . . . 20
- 3.4 Control alorithm implementation . . . 20
  - 3.4.1 Reaching loop process flow . . . 21

4 Methodology **24**
- 4.1 Goal broadcasting . . . 24
- 4.2 Task Priority . . . 25
  - 4.2.1 Task Description . . . 25

4.2.1.1 End Effector minimum altitude 25
4.2.1.2 Obstacle Avoidance 27
4.2.1.3 End Effector Target 29

5 Experiments **30**
5.1 30

6 Conclusions **31**

A Extra **32**

References **35**

List of Figures

2.1 Direct visual servoing scheme 4
2.2 Vision coordinate sistem 5
2.3 Obstacle avoidance schema 7
2.4 Dual-arm industrial robot example, SDA10 9
2.5 snake like robot from Crespi *et al.* (2005) 10
2.6 ANAT robot arm 11
2.7 Macro-Micro robot 12
2.8 Macro-Micro surgical robot 13

3.1 photo of the real system, inside the workcell 15
3.2 Simplified structure of the class JointRobotTP 16
3.3 Joint reaching control loop 19
3.4 Action server flow chart 20
3.5 Reaching loop flow chart 22

4.2 Activation for Obstacle avoidance 28

Chapter 1

Introduction

Robotic systems from their first introduction in the manufacturing field we relegated to work separated from the human workers. This was because the main use for robots was to perform, highly repetitive tasks, very fast or to work in dangerous environment. This removed the need to have interaction between human and robots. With the advent of industry "3.0" and "4.0" the focus shifted from that to have the robots collaborate with humans to increase efficiency, and to remove some burden from the human worker, especially for physically demanding tasks.

The concept of Human Robot Interaction (HRI) appears in the literature and can be divided in two broad categories, each with their respective challenges.

- **Physical Interaction**: interaction that require or could have some form of contacts with the robotic system.
- **Social Interaction**: interaction that aims to exchange information, or perform conversation of some kind.

In the context of this thesis, and more broadly in industrial applications, the focus is primarily on physical interaction. Collaborative robots operating alongside human workers must function in dynamic environments, where the human agent does not follow predefined trajectories.
As described in Weidemann *et al.* (2023), the **SestoSenso Project** proposes a framework for Human-Robot Collaboration in which controlled physical contact is not only possible but expected. Within this framework, the robot and the human operator jointly manipulate or work on the same object, requiring the robotic system to adapt continuously to

the human's actions. To support this type of collaboration, the robotic platform in the **SestoSenso Project** is equipped with proximity sensors that allow it to perceive changes in its surroundings and react autonomously and in real time. In addition, several robot links are covered with a sensorized tactile skin, enabling the system to detect and interpret physical contact with the environment or with the human collaborator. A key strength and novelty of the SestoSenso robotic setup lies in its multi-stage structure. The complete system features 12 degrees of freedom, created by combining two manipulators: a high-payload industrial arm from KUKA as the first stage, and a lightweight, highly compliant arm from Universal Robots as the second stage. This configuration allows the robot to leverage the strengths of both manipulators—power and precision from the KUKA arm, and flexibility and safety from the UR arm—making it well suited for collaborative tasks.

1.1 Research problem and objective

Since during the **SestoSenso Project** the two robots were controlled separately, in this work, I developed a unified control architecture with the aim to test the capabilities of the complete system in a series of experiments. Specifically with *reaching* and *obstacle avoidance* tasks. All the activities were carried out at MACLAB, the Mechatronics and Automatic Control Laboratory at Università degli Studi di Genova.

1.2 Thesis structure

After the brief introduction in 1 of the objective of this thesis, in 2 I will provide a literature review. 3 is the general description of the control architecture with a focus on the software implementation. In 4 instead the focus will be on the algorithms and methods I used in the architecture. Lastly 5 will be the presentation of the conducted experiments and the conclusion in 6.

Chapter 2

State of the art

In this chapter I will explore the literature on the topics of: *ambient perception* to expore different sensing approaches to control manipulators in a dynamic environment. I will than focus my attention on the uses of this sensory information in the task of *obstacle avoidance* During my state of the art research it was apparent a lack of literature on the specific architecture developed in the *SestoSenso project*. For this reason the last part of this chapter has focused more broadly on *high dof architecure*, to explore how long open kinematic chain system are treated, and in which areas are used.

2.1 Environment perception and awareness

Environment perception is one of the biggest difference when we move from the classical use of robotic systems in industry, to a more modern framework geared towards HRI. In this section I reported two of the main method for extracting ambient morphology information from various types of sensors, and explained their strength and weaknesses.

2.1.1 Image recognition based methods

As reported in Badrloo *et al.* (2022), we can divide vision based methods in two main categories:

- Monocular vision: use a single camera mounted on top of the robot.
- Stereo vision: use two synchronized cameras.

The basic approach of the *visual servoing* with monocular camera can be represented in the following schema:

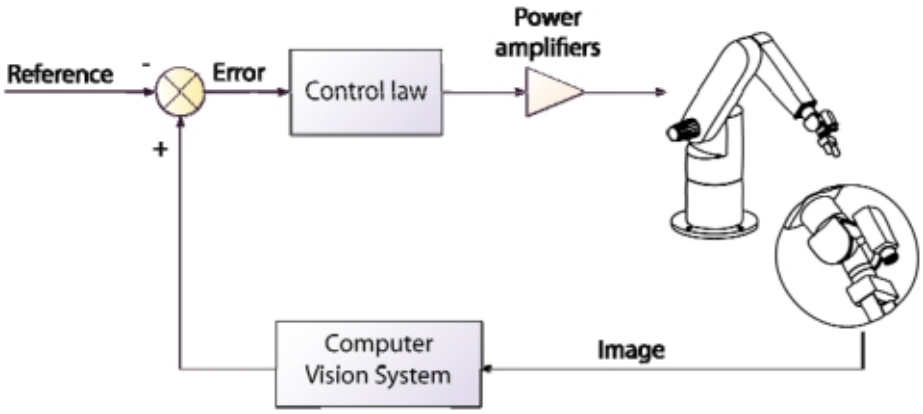


Figure 2.1: Direct visual servoing scheme

In the work of Muslikhin *et al.* (2020) we can see how a monocular system is used with a *deep Region based Convolutional Neural Network* to recognize objects in the field of view of the robot and decide if said object is a target or not. This first step is than followed by a *kNN* and the *Fuzzy interference system* to localize 2D position of the targets, the last coordinate is found by only shifting the end-effector a few millimeters towards the x-axis.

Following and improving the capabilities of a singular camera system there is the use of: stereo vision. Stereo vision works by combining the information of the two cameras, that are placed in a known position to extract information of the third dimension of objects in the images. In the work of Huh *et al.* (2008) we can see how a stereo vision based system is used to perform obstacle recognition on a autonomous driving vehicle.

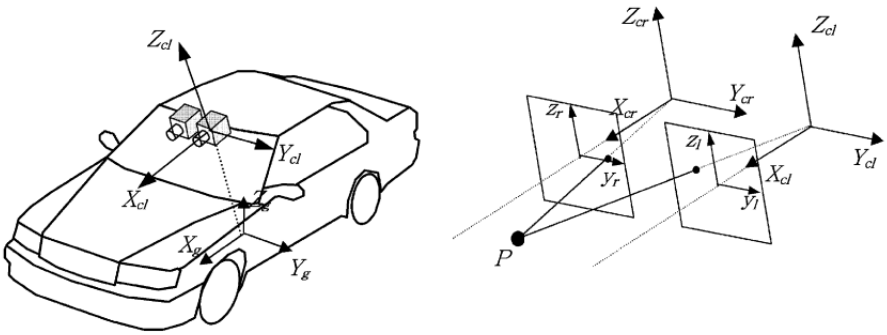


Figure 2.2: Vision coordinate sistem

The image based methods in general appear to have some key characteristics that make them impractical for effective *obstacle perception* in an environment such as the one of the **SestoSenso project**.

With the advent of AI in recent years the use and potential of *image recognition* and with it all the vision based systems has greately increased. But they still have a series of weaknesses that have a large impact when it comes to develop viable systems for industrial application. In the work of Panasiuk (2025) a control system using a 3d stereo camera and the YOLO AI algorithm for image recognition, those limitation are evident:

- **Hardware and software**: The cost of the system parts is not paltry, form the camera to the AI software.
- **Integration and configuration**: The camera apparatus has an implementation that is task-specific, which means that every change in the environment requires a complete re-configuration of the system.
- **Computational cost**: The image analysis is performed on a separated computer to manage the burden of the computation.
- **Field of view**: The camera visualize only the work area, which is not adequate for a HRI situation.
- **Environment interference**: The use of 2D and 3D image information, require to have a strict control on the occlusion and disturbances in the environment, from lighting to airborne dust. This level of control is not possible in a industrial context.

- **Privacy**: One problem not addressed by the paper is the privacy of people working around or with the robot, that is not mantained with the use of a camera.

2.1.2 Point-cloud discretization based methods

In the work of Zauner *et al.* (2025) three different type of spatial perception sensor are evaluated to create point cloud of a robot's workspace. To perform safe navigation and avoid collisions. the sensor used are:

- **Time Of Flight**: *Kinect V2* and *Omron OS32C Lidar*
- **Active Stereoscopy**: *Intel RealSense D435*

The two time of flight sensor work with a infrared light and a laser respectively and the measure the distance from an object by timing the time delta at the reception of the light impulse. The Intel sensor instead is based on the *stereo vision* principle, but it uses simpler cameras aided by a infrared projector that imposes a grid of points onto the surfaces. The sensors are mounted on the *end effector* of the manipulator and panned over the workspace to record a sample of the environment, the resulting pointcloud is than processed to reduce the number of points and to extract feature of the environment.
In the paper the extracted feature are used to simplify the 3D representation of the obstacle, and to perform collision-checks they confronted a series of different algorithms. In the case of my thesis I stopped after the filtering to reduce the number of points, than the point cloud is directly used to represent the robot and obstacle in the simulation.
As stated in Hussmann *et al.* (2008) the main drawback of *ToF* sensors is the lower resolution capabilities in comparison to *stereo vision* techniques. The paper highlight that even with this performance deficit the *ToF* were viable to be used in automotive application even for safety tasks.

2.2 Obstacle avoidance in HRC

For *Human–Robot Collaboration* applications, the robot must operate under a *multi-objective* control strategy, where the system handles a *goal-driven task* defining the role of the robot, and one or more other task that go from safety to optimization tasks. Within collaborative

scenarios, the safety layer must be treated with **higher priority**, temporarily overriding the main objective whenever a hazardous situation is detected, to guarantee human and system protection in real time. In this thesis, the prioritized secondary objective ensuring safe collaboration is *obstacle avoidance*, which monitors the robot surroundings and generates motion corrections when the robot is to close to the obstacle.

In the case of a manipulator arm we have to also include the z axis, since we are operating in 3d space. Looking at the work of Zhang & Sobh (2003) we can see how we can compute a safe trajectory for a *SIR-1* robot manipulator using cubic polynomials for a path with intermediate points. In this paper is interesting the introduction of the concept:*link* collision avoidance. By controlling the *link* closest point to the obstacle and using the analytical formulation of the *Inverse Kinematics* and the *obstacle shape* it is possible to define *joint variables* constraints to ensure a collision free navigation.
In the work of: Maciejewski & Klein (1985) instead the proposed approach considers the closest point of the hole robot to the obstacle, than it apply a velocity vector to said point that is directly opposite to the distance vector $(P_{ob} - P_{rb})$.

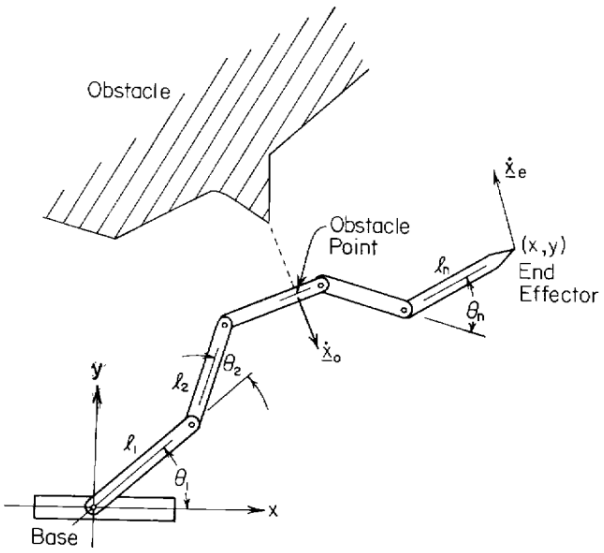


Figure 2.3: Obstacle avoidance schema

To ensure that the *obstacle avoidance* desired velocity does not impact the tracking of the *end effector velocity* the joint space velocity are searched in the null-space of the solution to the first problem. This ap-

proach allows to leverage the redundancy of the manipulator. The proposed algorithm is than applied to a planar robot with parallel *revolute* joints, as shown in the image. And also to a 3D redundant manipulator operating through the window of an automobile door.

2.2.1 Redundancy control

From the discussion of the previous section it is clear that to correctly perform obstacle avoidance the control architecture has to deal with multiple objectives. This objective need to be *task oriented* to allow for portability between different system configuration. The classic framework for this type of control was developed by Slotine & Siciliano (1991) and extended by Simetti & Casalino (2016) to include the activation and deactivation of task without discontinuities. The general idea is to have a *hierarchy of tasks*, defined to be *objective specific* and not connected to the particular structure of the robot. Given a generic objective function defined in the task space:

$$\dot{\boldsymbol{x}}_i = \boldsymbol{J}_i(q) \cdot \dot{\boldsymbol{q}} \tag{2.1}$$

- $\dot{\boldsymbol{q}} \in \mathbb{R}^{(n\times 1)}$: joint displacement vector.
- $\dot{\boldsymbol{x}}_i \in \mathbb{R}^{(m_i\times 1)}$: task velocity vector, or *reference rate*.
- $\boldsymbol{J}_i(q) \in \mathbb{R}^{(m_i\times n)}$: task jacobian matrix.

Given that the solution of the highest priority task is:
$\dot{\boldsymbol{q}}_1 = \boldsymbol{J}_1^{\#}\dot{\boldsymbol{x}}_1 + (\boldsymbol{I} - \boldsymbol{J}_1^{\#}\boldsymbol{J}_1)\dot{\boldsymbol{z}}, \forall \dot{\boldsymbol{z}}$ the second part of the solution is the projector on the *null space* of $\boldsymbol{J}_1$, the solution to the lower priority task are searched in that space. Yielding the general solution:

$$\dot{\boldsymbol{q}}_i = \dot{\boldsymbol{q}}_{i-1} + \boldsymbol{J}_i(\boldsymbol{I} - \boldsymbol{J}_i^{\#}\boldsymbol{J}_i)(\dot{\boldsymbol{x}}_i - \boldsymbol{J}_i\dot{\boldsymbol{q}}_{i-1}) \tag{2.2}$$

In the paper is demonstrated that the solution of a lower priority task does not modify the higher one, but it is *attempted* in the null space.

2.3 High DoF architecture

In this section I want to explore some of the relevant high-dof architecture found in the literature.

2.3.1 Dual-arm systems

In recent years there has been a trend to use these dual-arm systems for HRC(Human Robot Collaboration), but also for replacing human workers without the need to redesign the work cell.

Figure 2.4: Dual-arm industrial robot example, SDA10

As is stated in the survey of Smith *et al.* (2012) the strengths of the dual arm architecture are:

- *Similarty to operator*: useful both in the case of HRC and to substitute the human worker with minimal effort.
- *Flexibility and stiffness*: Combining the stiffness of closed chain manipulation, with the flexibility of a serial link.
- *Manipulability*: High number of DoFs allows for complex motion tasks.
- *Cognitive motivation*: The similar charcteristics of the kinematic chain is belived to be helpful in HRC context.

In most cases for these architectures the *obstacle avoidance* is computed for the *navigation* if the robot has a movable base. Moreover the interaction with the environment is performed with the use of *visual servoing*, witch was firstly discussed by Hutchinson *et al.* (2002), position based and *hybrid* methods, combining visual and position information.

2.3.2 Snake-like robot

A completely different class of robot is represented by the "*snake like*" robot. As shown in the work of Hirose & Yamada (2009) and Crespi *et al.* (2005) these types of robot are biologically inspired, and they can produce a forward motion from an undulatory one. Reproducing the movement patterns of snakes.

Figure 2.5: snake like robot from Crespi *et al.* (2005)

The potential of these robot that are currently being explored are for navigation in tight spaces, to be applied to endoscopes for examples. In addition the interest lies in the flexibility of a "*snake like*" body, since it could be used to move, climb and grasp if needed.

2.3.3 Planar robot

For more industrial application, I reviewed the work of Le Boudec *et al.* (2006) and Maciejewski AA (1985). These work take into consideration high-dof planar manipulator, and they also propose two approaches to do *Obstacle avoidance* with their respective architecture. In the case of Le Boudec *et al.* (2006) the paper uses the *ANAT* robot, presented in the figure below.

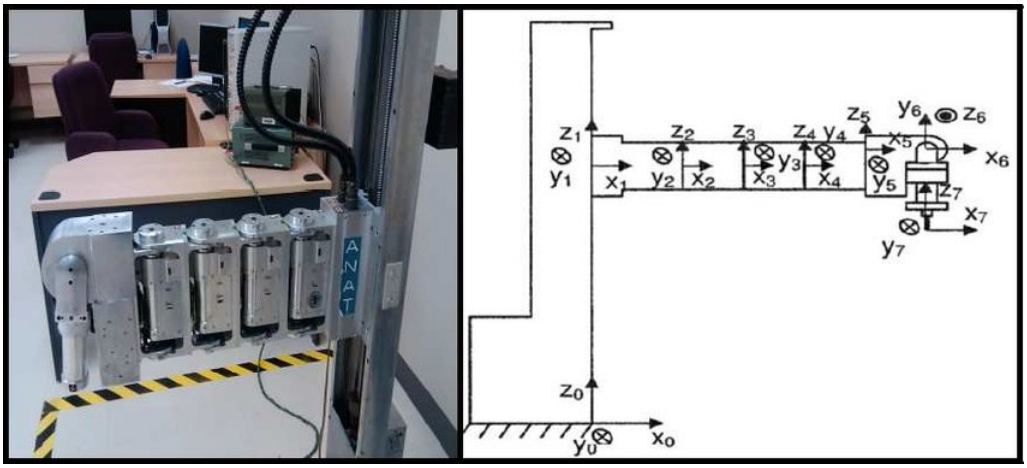


Figure 2.6: ANAT robot arm

This is a 7Dof robot, comprised of 1 *prismatic* joint to control the z cordinate, followed by 3 parallel *revolute* joints and a 3 Dof *wrist*.
The proposed control algorithm is based on the work of Zlajpah (1997) for the computation of the generalized inverse of the jacobian matrix. In addition, the obstacles are modeled as hyper planes to reduce computational costs and the control law is applied to the joints in order. In this paper the proposed method is applied at the *Dynamic* level, the objective function for the *Obstacle avoidance* is computed as follows:

$$V_1(q) = \sum_{i=1}^{m} \sum_{j=2}^{n} \frac{\alpha_{ij}}{-(\frac{x_j - x_{ci}}{r_i + r_{si}})^2 - (\frac{y_j - y_{ci}}{r_i + r_{si}})^2 - (\frac{z_j}{h_i + h_{si}})^2 + 1} \tag{2.3}$$

where:

- *m*,*n*: respectively the number of obstacles, and the number of points placed on the robot.
- α_{ij} : weight of the constraint for joint *i* from obstacle *j*
- (x_j, y_j, z_j) : coordinates of joint *j* in the *base frame*
- $(x_{ci}, y_{ci}, r_i, h_i)$: coordinates of cylinder *i* in the *base frame*
- (r_{si}, h_{si}) : safety distances in *radius* and *height* from cylinder *j*

The approach generates a *repulsive force* that becomes stronger as the robot approaches an obstacle. While *potential-field techniques* are a standard choice for *dynamic obstacle-avoidance control*, I could not adopt

them in this work because the robots' *dynamic controllers* were locked behind the manufacturer's proprietary software, preventing access to the required control layer.

2.3.4 Macro Micro configuration

The last interesting configuration I want to talk about is referred in the literature as *Macro-Micro Robot*. Firstly proposed by Sharon & Hardt (1984), the objectives of the proposed architecture were to resolve the opposing problems of *speed* and *tracking precision* and also to correct the errors in *end point measurement*, given by bending in the links and errors in the measurement errors in the encoders.

The uses and capabilities of this configuration are presented in the work of Zhou *et al.* (2022), following is a photo of the robot they used.

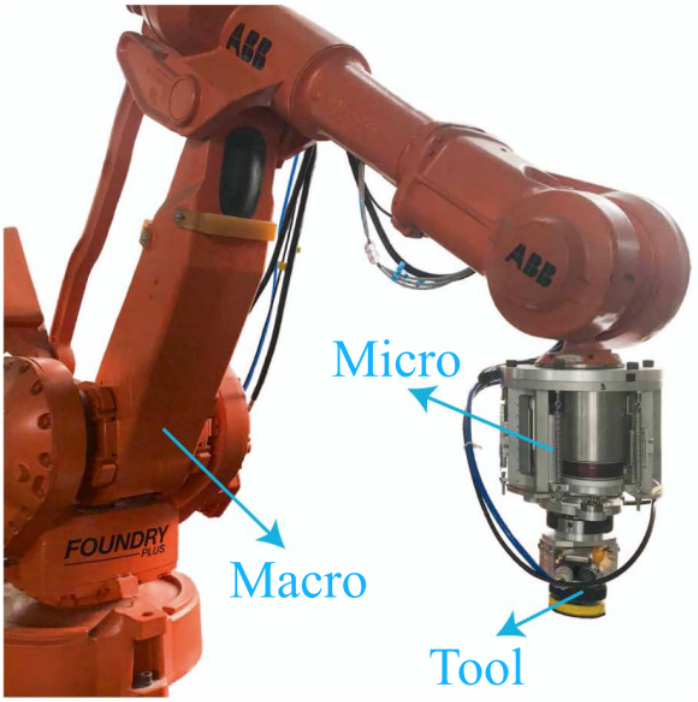


Figure 2.7: Macro-Micro robot

The robot is composed of a 6 Dof *Macro* manipulator and a 3 Dof *Micro*. The robot is equipped to perform polishing tasks on complex

surfaces.
The paper focus is to prove the effectivenss of a *sampling based* motion assignment(MA) strategy with multi performance optimization. Since the robot has a total of 9 degree of freedom there is space for optimization in the robot movement. The configuration optimization function is to be minimized for each sampling point of the chosen trajectory, and for each point the performance index and constraint ($RPI_{c,i}(q)$) must be computed. Classic *gradient based* methods can easly stop at local minima since the function is not convex, the proposed MA aims to optimize the movement of macro and micro manipulator, avoiding the costly and error-prone computation. The system on witch I worked on this thesis is of the same general structure, but the two robot are considered as a whole. Also in my work I am not computing any offline trajectory as in the case of this paper.
Another application of the *Macro-Micro* configuration is in the field of medical robotics, in the work of Cursi *et al.* (2022). In this paper the proposed architecture is composed of a *KUKA LBR IIWA* robotic arm with 7 Dof, and a *Micro-IGES* surgical robotic instrument with 7 Dof(2 Dof are composed of the jaws of the instrument).

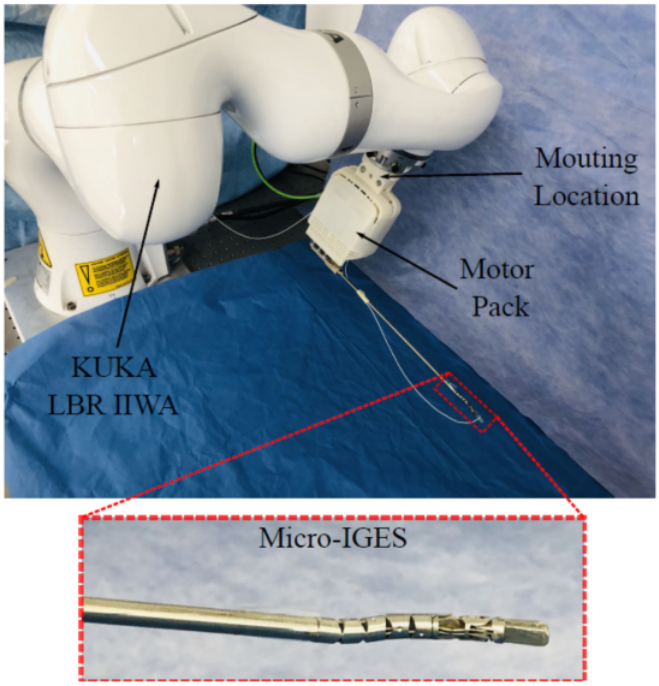


Figure 2.8: Macro-Micro surgical robot

In this paper the objective was to demonstrate that the overall performance of the system can be improved by defining preoperatively the best initial configuration of the surgical instrument in terms of *roll*, *pitch* and *yaw* with respect to the macro serial-link manipulator to achive maximum accuracy in performing specified tasks. The paper highlights how the macro micro manipulator configuration allows for completion of multiple-objective tasks, such as:

- *Guarantee Remote Center of Motion*: The RCM(which for surgical application is usually the incision site) has to remain stationary.
- *Desired path tracking*
- *Assembly errors compensation*

The method used in this paper starts with a *Genetic algorithm* used to generate possible configuration, that are than evaluated through *Hierarchical Quadratic Programming*. The solution of the procedure finds the best intial configuration based on a fitness function and resiliance to errors.

Chapter 3

Architecture implementation

3.1 System Description

The robotic system I worked with was composed of two articulated industrial robot, namely a **Kuka KR150** from *Kuka* and a **UR10e** form *Universal Robot*.

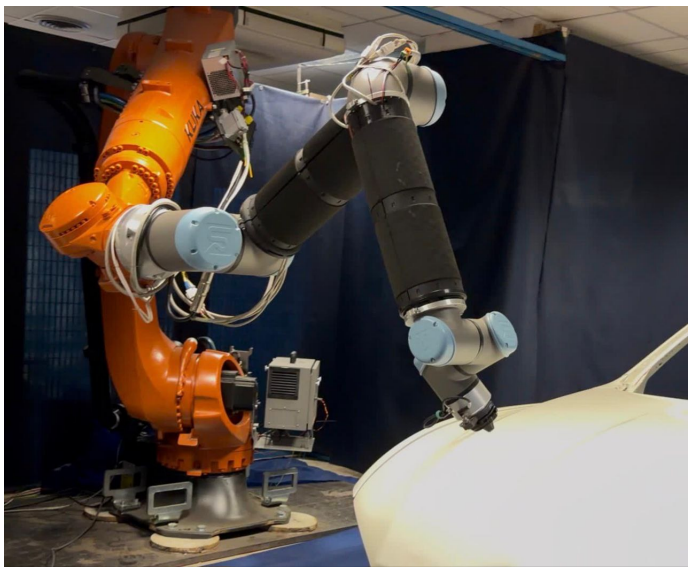


Figure 3.1: photo of the real system, inside the workcell

The simulation part I used to develop my architecture was composed of the classes: `KukaRobot` and `UR10eRobot` . These two classes themselves inherits from the class: `RobotInterface`.
All of the code was developed usinc `c++` language, and the library `Eigen` to handle matrices.

The parent class contains the information for:

- *Transformation* using a `tf2_ros::Buffer` and `tf2_ros::TransformListener`.
- *Robot state*, stored as a vector of joint variables q and velocities $\dot{q}$ `Eigen::VectorXd`.
- *Inverse kinematics* using `KDL::Tree` and `Chain`, a data structure to recursively compute the jacobians of the robots. Starting from the `urdf` description of the robots.

The transformation buffer contain the kinematic chain of the two robots, as `geometry_msgs::msg::TransformStamped`. These transformation are periodically updated trough a topic, published by the *robot state publisher*. The `tf2_ros::Buffer` also allows to retrive specific frame to frame trasformation, using the frames id. Than using `KDL` the inverse kinematics can be computed on demand during the movement of the robot.
My work was the development of the class: `JointRobotTP` that takes these two classes and combines them in an unified architecture.

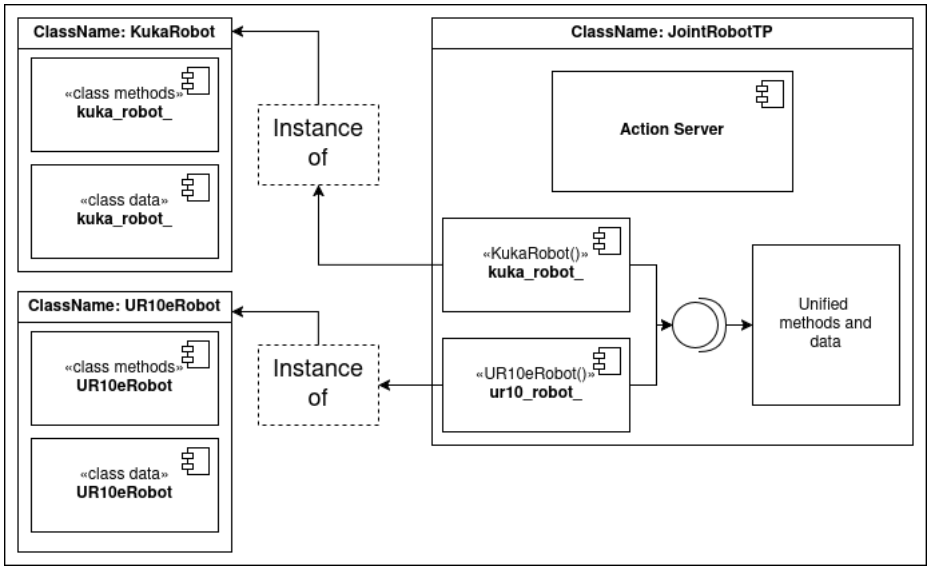


Figure 3.2: Simplified structure of the class JointRobotTP

The main goals I wanted to pursue were the following:

1. Have an efficient way to send commands to the unified robot.

2. Use a data structure that allows for easy addition and removal of tasks from the hierarchy.

3. Keep a degree of separation between the robot information and the control algorithm.

In the following section I will describe more in detail the structure of the *JointRobotTP* class.

3.2 JointRobotTP Class Implementation

The unified robot class implementation main *methods* and *data structure* are:

Class setup	`Initialize([Args..]);`
	`insertInitConfigMap();`
	`insertFuncPointerVtc();`

Data structure	`map<string, VectorXd> initial_configurations_map_;`
	`vector<ProximityTask> proximity_task_points_;`
	`map<string, tp_task> TP_task_map_;`

Task state update	`Update_TRR_<task name>();`
	`Update_AFunc_<task name>();`
	`Update_TskJac_<task name>();`

Action server	`execute([Args..]);`
Robot movement	`ReachInitialConfiguration([Args..]);` `RunCartesianReachingLoop([Args..]);`

3.2.1 Class setup

These methods are created to instantiate the class object once the executable is started.
Specifically the method `Initialize()` is tasked with the creation of the nodes for the robots classes and all the other part of the communication structure. It takes as arguments three node pointers that are used to instantiate one object of the class `KukaRobot`, `UR10eRobot` and `JointRobotTP`.
It creates a subscriber to the `proximity_task` topic and the action server for the robot movent. Lastly the other two initialization methods

are called, the first `insertInitConfigMap()` inserts all the initial configuration vector in the corresponding data strucuture (`map<string, VectorXd> initial_configurations_map_;`), and the second is used to create three function pointers vector that are used to call all the *task state update methods* during the movement of the robot.
This structure was chosen to have a more manageable code and to accomplish the second objective 3 I set for the architecture.

3.2.2 Data structure

For the control algorithm I had to define three different matrices for each task I created. I decided to implement a `struct` to store the matrices and put everything in a map(`map<string, tp_task> TP_task_map_;`). The `struct tp_task` is defined as follows:

```
Eigen::MatrixXd RefRate;
Eigen::MatrixXd ActMatrix;
Eigen::MatrixXd TskJacobian;
```

This definition uses dynamically sized matrices that are useful in the case of the **task priority** control since the dimension of each matrix can change from task to task(maintaining a certain relation within each task *RefRate* $\in \mathbb{R}^{(m\times 1)}$; *ActMatrix* $\in \mathbb{R}^{(m\times m)}$; *TskJacobian* $\in \mathbb{R}^{(m\times n)}$). The use of `std::map` also allows to reference to each task trough a `string` witch is a very flexible and efficient approach(search complexity $O(\log n)$, with n the element number in the map).

3.2.3 Task state update

Just to explain the logic in my implementation, for each task defined for a particular objective in the control algorithm I developed three different methods used to cyclically update the information contained in `map<string, tp_task> TP_task_map_;`.
Than the pointers to these methods are inserted in a vector, and this vector is used to call all the function in a loop. This is functional since I can remove a task from the update cycle just by commenting three rows, and I have a clear idea of which task I am updating in a very concise way.
The specific computation needed of each task will be discussed in chap.4

3.3 Action server

The main part of the action server is the method `execute([Args..]);` aside from the communication related methods needed to create the channel in the *ROS2* framework. This method takes as argument the a shared pointer to the `goal_handle` of the current active goal.
The methods handles the reception of the goals and the publication of the *goal frame*, projected in the correct *reference frame*. Than it uses the information in the *goal message* to select the desired intial configuration that is reached using `ReachInitialConfiguration([Args..]);` selecting a vecotr from
`map<string, VectorXd> initial_configurations_map_;` and sent to the *Kuka* and *UR10e*, the initial vector has 12 elements, that are splitted and set to their respective robot. A simple control loop is than used to control the joint:

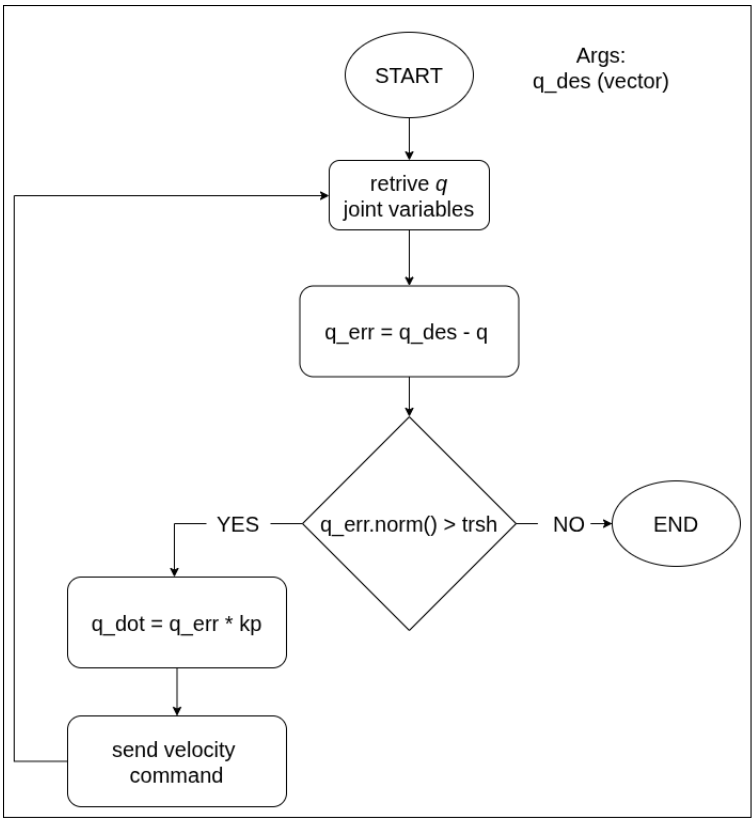


Figure 3.3: Joint reaching control loop

the loop drives each joint towards the desired configuration. The integration of this part in my architecture was done improve repeatability, to have the possibility to impose a common starting position in different experiments.

3.3.1 Action Server process flow

The core functionalities of the `execute();` method are here represented in a flow-chart to highlight better the loop inner workings:

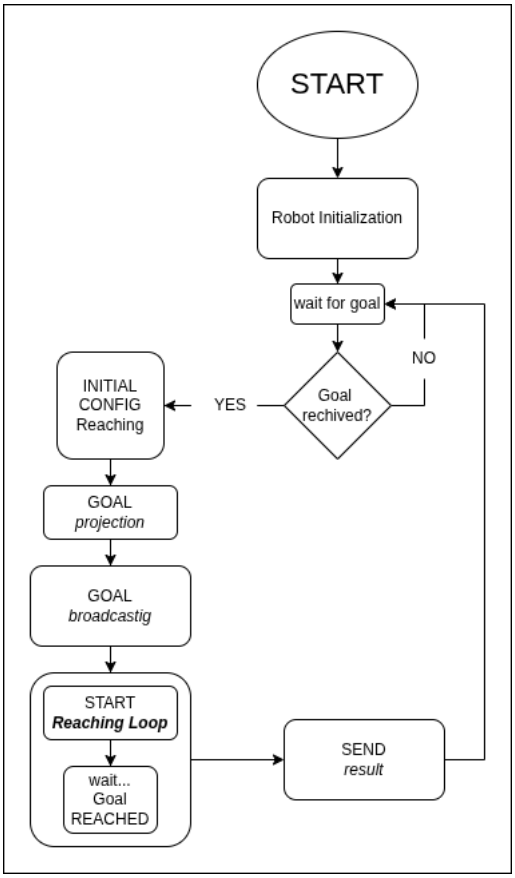


Figure 3.4: Action server flow chart

3.4 Control alorithm implementation

The structure of the control algorithm was derived from the form of the *task state* data structure, I wanted to have in the control loop a very

clear way to identify which tasks I was activating. And to *passively* impose the priority without needing to use a flag.
With this in mind I created a class called: `TPComputation`, of witch I could instantiate an object for each iteration of the reaching loop, the class data would be the *null space* of the *task jacobian* computed at the previous step, Q_{k-1}, and the resulting *joint velocity* vector, ρ_{k-1}. These data would be extracted from the class as needed to send the command to the robot.

Instance handling	`init_TPComputation ([Args...]);` `kill_TPComputation ();`
Step computation	`computeTP_step ([Args...]);`
Matrix inversion	`REG_Pinv_operator ([Args...]);` `REG_Pinv ([Args...]);`
Data extraction	`getTP_ydot ();` `getTP_Q ();`

To initialize the class object the method `init_TPComputation ([Args...]);` takes as arguments the number of *Dof* of the structure, used to initialize the dimension of the internal matrices and some parameter that are used in the pseudo-inversion of the *augmented jacobian*. Since I wanted to be sure that no information remained after any cycle, I also added the method `kill_TPComputation ();` that *clears* the internal variables after use.
The step computation of the *task priority inverse kinematics* is performed using the `computeTP_step ([Args...]);` method. The arguments taken by the function are the matrices of the desired task, stored in `map<string, tp_task> TP_task_map_;`.
Internally the computation is than performed using also:
`REG_Pinv_operator ([Args...]);` and `REG_Pinv ([Args...]);` but I will better discuss the technique used in chap. 4. Lastly after all the desired steps have been computed, the resulting *joint velocity vector* is extracted using: `getTP_ydot ();`.

3.4.1 Reaching loop process flow

To better show the intended and actual use of the `TPComputation` class to control the robot, I added the flow-chart for the `RunCartesian-ReachingLoop([Args..]);` method.

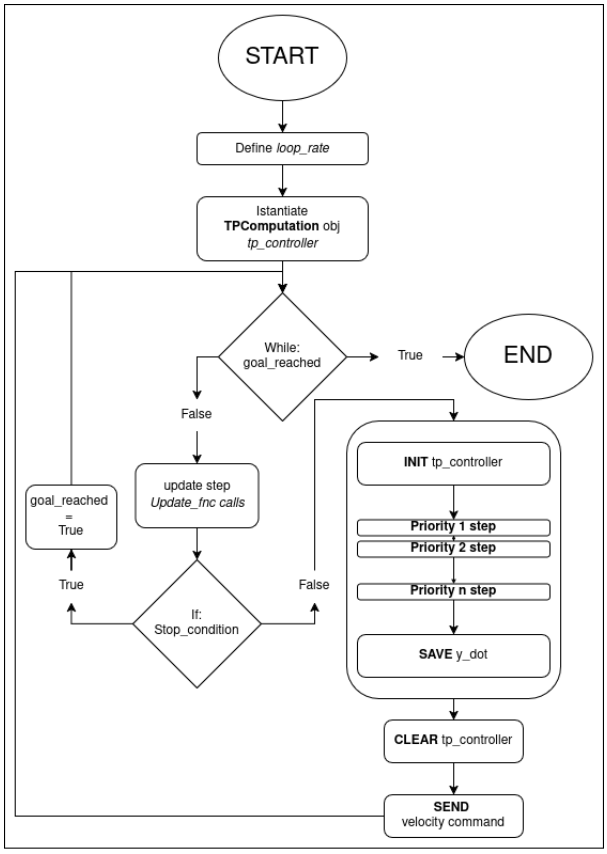


Figure 3.5: Reaching loop flow chart

To briefly describe the loop, after the instantiation of the `TPComputation` object the loop starts. In the *Update step*, all the function that are relative to a *task* are called, starting with the *Reference Rate*, *Activation function* and lastly *Task Jacobian*. The second part, the *Stop condition* is checked. If the control is positive the loop is immediately stopped, and the result is sent to the action server client. if the opposite is true, the *tp_controller* is initialized, with the *Dof* of the system and the variables for the pseudo-inverse computation. Than each "*step*" of the algorithm is performed, finishing with a "*null*" task, composed of

two Identity matrix for *Activation function* and *Task Jacobian*, and with a zero vector for *Reference Rate*. The loop is repeated until the stop condition is met or stopped after a time limit, in this case the goal is considered not reached.

Chapter 4

Methodology

in this chapter I will describe in detail the mathematical computation performed by the control algorithm I developed.

4.1 Goal broadcasting

The architecture works in a *position reaching* framework, a goal position for the *end effector* is sent to the robot and the algorithm tries to reach said position, with constraint given by other tasks. The goal is sent using the *action server* build in the *JointRobotTP* class, in the form of two vectors:

$$\boldsymbol{r} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad ; \quad \boldsymbol{\rho} = \begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix} \tag{4.1}$$

The first represent the desired translation, and the desired rotation as *roll, pitch, yaw* angles. The projection frames of this vector to define the goal position and orientation could be either the *end effector* frame, or, the *kuka base* frame. This was done for experimental purposes, for easly sequencing different goals, or to repeat reaching tasks to a specific position in the environment.

For the orientation of the goal, $\boldsymbol{\rho}$ is used to compose the rotation matrix that is than projected on the desired frame:

$$\boldsymbol{R}_{goal} = \boldsymbol{R}_z(\psi) \cdot \boldsymbol{R}_y(\theta) \cdot \boldsymbol{R}_x(\phi) \tag{4.2}$$

$$\langle \textit{kuka\_base} \rangle : {}^{kb}\boldsymbol{R}_{goal} = \boldsymbol{I} \cdot \boldsymbol{R}_{goal} \tag{4.3}$$

$$\langle \textit{end\_effector} \rangle : {}^{ee}\boldsymbol{R}_{goal} = \boldsymbol{R}^{kb}_{ee} \cdot \boldsymbol{R}_{goal} \tag{4.4}$$

In the first case the projection matrix is the identity since $\langle kuka\_base \rangle \equiv \langle world \rangle$. Same process is done for the translation vector:

$$\langle kuka\_base \rangle : {}^{kb}\boldsymbol{r}_{goal} = \mathbf{0} + \boldsymbol{r}_{goal} \tag{4.5}$$

$$\langle end\_effector \rangle : {}^{ee}\boldsymbol{r}_{goal} = \boldsymbol{r}_{ee}^{kb} + \boldsymbol{r}_{goal} \tag{4.6}$$

Than the rotation matrix and the translation vector are use to publish the $\langle goal \rangle$ in rviz.

4.2 Task Priority

Continuing the discussion from 2.2, the classic task priority algorithm as explained in Simetti & Casalino (2016) lacks the ability to smoothly activate and deactivate *inequality* task when the robot is far from the activation region. The approach that I implemented is based on the definition of a new *regularized pseudo-inversion operator* that integrates the *activation function* as a weight matrix to modulate the intensity of the action taken for a specific task. The operator is defined as:

$$\boldsymbol{X}^{\#,\boldsymbol{A},\boldsymbol{Q}} \triangleq (\boldsymbol{X}^T\boldsymbol{A}\boldsymbol{X} + \eta(\boldsymbol{I}-\boldsymbol{Q})^T(\boldsymbol{I}-\boldsymbol{Q}) + \boldsymbol{V}^T\boldsymbol{P}\boldsymbol{V})^{\#}\boldsymbol{X}^T\boldsymbol{A}\boldsymbol{A} \tag{4.7}$$

where the matrix $\boldsymbol{V}$ is the right orthonormal matrix of the SVD decomposition for $\boldsymbol{X}^T\boldsymbol{A}\boldsymbol{X} + \eta(\boldsymbol{I}-\boldsymbol{Q})^T(\boldsymbol{I}-\boldsymbol{Q})$. The compact expression of the algorithm becomes, for the *k-th* priority level:

$$\begin{aligned}
\boldsymbol{W}_k &= \boldsymbol{J}_k\boldsymbol{Q}_{k-1}(\boldsymbol{J}_k\boldsymbol{Q}_{k-1})^{\#,\boldsymbol{A}_k,\boldsymbol{Q}_{k-1}} \\
\boldsymbol{Q}_k &= \boldsymbol{Q}_{k-1}(\boldsymbol{I} - (\boldsymbol{J}_k\boldsymbol{Q}_{k-1})^{\#,\boldsymbol{A}_k,\boldsymbol{I}}\boldsymbol{J}_k\boldsymbol{Q}_{k-1}) \\
\boldsymbol{\rho}_k &= \boldsymbol{\rho}_{k-1} + \boldsymbol{Q}_{k-1}(\boldsymbol{J}_k\boldsymbol{Q}_{k-1})^{\#,\boldsymbol{A}_k,\boldsymbol{I}}\boldsymbol{W}_k(\dot{\boldsymbol{x}} - \boldsymbol{J}_k\boldsymbol{\rho}_{k-1})
\end{aligned} \tag{4.8}$$

4.2.1 Task Description

In this section I will describe the mathematical formulation of the task I implemented in my control loop. To simplify the notation all the matrices, if not stated otherwise, are projected on the $\langle kuka\_base \rangle$ reference frame.

4.2.1.1 End Effector minimum altitude

This task is used to keep the end effector away from the floor.

Task reference:
The reference rate for this task is computed using the z coordinate of the traslation from the $\langle$*kuka_base*$\rangle$ frame to $\langle$*end_effector*$\rangle$.

$$\textit{Reference rate: } \dot{\bar{x}}_z = \lambda \cdot (\bar{\boldsymbol{x}} + \delta - \boldsymbol{z}_{ee}^{kb}) \tag{4.9}$$

The complete vector has all zero exept on the z coordinate.

$$\textit{Reference rate: } \dot{\bar{\boldsymbol{x}}} = [0, 0, \dot{\bar{x}}_z, 0, 0, 0]^T \tag{4.10}$$

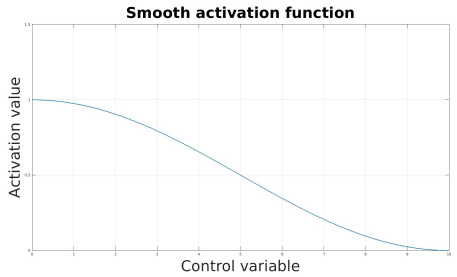
The *Reference rate* is considered as the *desired shape* of the derivative of the variable I want to control. To be more precise:

- if $x > \bar{x}$ then $\dot{x} < 0$
- if $x < \bar{x}$ then $\dot{x} > 0$
- if $x = \bar{x}$ then $\dot{x} = 0$

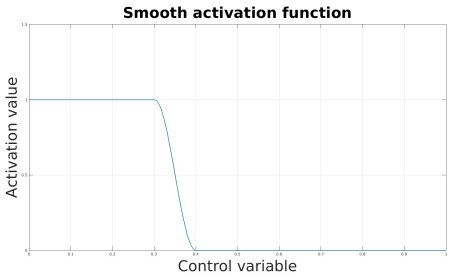
In the case of the *Minimual altitude task* the variable I want to control is the z of the *end effector* of the robot, and to impose a minimum altitude z_{min}. Finally the δ keeps count of the activation region, meaning that the task will be smoothly activated starting from $z_{min} + \delta$ to z_{min}.

Activation function:
To avoid the abrupt activation of the task that could lead to chattering around the treshold, and more importantly to avoid the over constraining of the system we also add a smooth *activation function*.

$$a = \begin{cases} 1 & x < x_{min} \\ \frac{1}{2}[(\cos(\pi \frac{x - x_{min}}{\delta}) + 1)] & x \in [x_{min}, x_{min} + \delta] \\ 0 & x > x_{min} + \delta \end{cases} \tag{4.11}$$



(a) Generic function



(b) With realistic values

In this case I used a (6×6) zero matrix as my activation function, than I inserted the value in the third position of the major diagonal, ending up with:

$$\boldsymbol{A}_{minalt} = \boldsymbol{diag}(0, 0, a_z, 0, 0, 0) \tag{4.12}$$

using $z_min = 0,30m$ as my lower limit and $\delta = 0,1m$, and the function is tuned to yield $y \in [0, 1]$.

Task Jacobian
For this task the jacobian matrix is equivalent to the geometric jacobian, computed up to the *end effector* frame. Since the available methods of the classes `KukaRobot` and `UR10eRobot` compute the jacobians matrices in their respective base frames. Enumeratin the links of the whole robot as $lk_i = 1...12$

$$^{kb}\boldsymbol{J}_{6/kb}; \, ^{ub}\boldsymbol{J}_{12/ub} \tag{4.13}$$

The complete trasformation for computing the Jacobian of the unified structure projected in the kuka base frame is:

$$content... \tag{4.14}$$

I decided to build the jacobian colum by colum since it was necessary for the obstacle avoidance task and it allowed for a more clear and concise code. So for computing the colum relative to the $i-th$ link I used the equation:

$$^{kb}\boldsymbol{J}_{i/kb} = \begin{pmatrix} ^{kb}\boldsymbol{J}^{L}_{i/kb} \\ ^{kb}\boldsymbol{J}^{A}_{i/kb} \end{pmatrix} = \begin{pmatrix} \boldsymbol{R}^{kb}_{i} \cdot \boldsymbol{ax}_i \times (r^{kb}_{n} - r^{kb}_{i}) \\ \boldsymbol{R}^{kb}_{i} \cdot \boldsymbol{ax}_i \end{pmatrix} \tag{4.15}$$

where $\boldsymbol{ax}_i$ is the axes of rotation for joint i expressed in the joint frame, which I retrieved from the `urdf` description of the two robots.

4.2.1.2 Obstacle Avoidance

For the *Obstacle Avoidance* task I used the approach proposed in Maciejewski & Klein (1985) keeping a single point task were that point is the point at minimum distance from the obstacle in the robot body.
Task reference:
As a task reference I used the opposite of the vector connecting the *minimum distance* robot point to the *obstacle point*.

$$d = (P_o - P_r) \quad ; \quad \dot{\bar{\boldsymbol{r}}} = -d \tag{4.16}$$

In chap.5 I will experiment with both the vector as is, and with a zeroed out z component. To understand how the proposed approach behaves in the 3d and 2d case.

Activation function:
The activation function is a 3×3 matrix, and the values are computed in the same fashion as the *minimum altitude* task. This time the minimum distance is lowered to 10cm while $\delta = 50$cm, this was to enforce a very smooth activation of the task. The resulting profile is as follows:

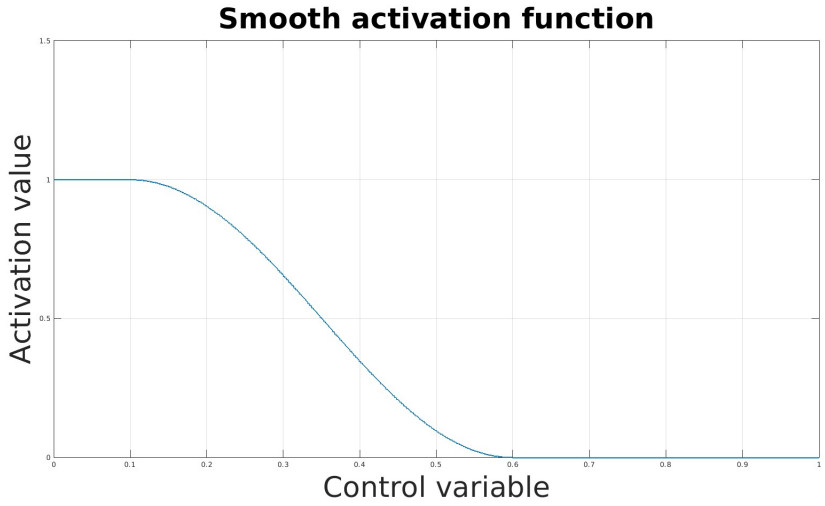


Figure 4.2: Activation for Obstacle avoidance

Task Jacobian:
For the task jacobian I used the same analytical approach as in the *minimum altitude* task, adding a rigid body to take into consideration the position of the *minimum distance point* on the body of the robot. Taken the complete jacobian of the 12Dof structure I pre-multiplied for the rigid body jacobian, which is computed as:

$$^{kb}\boldsymbol{J}_{mdp/kb} = \begin{pmatrix} ^{kb}\boldsymbol{J}^{L}_{mdp/kb} \\ ^{kb}\boldsymbol{J}^{A}_{mdp/kb} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{3\times3} & [r_{mdp/i}\times]^T \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} \end{pmatrix} \cdot \begin{pmatrix} ^{kb}\boldsymbol{J}^{L}_{i/kb} \\ ^{kb}\boldsymbol{J}^{A}_{i/kb} \end{pmatrix} \tag{4.17}$$

mdp:*minimum distance point*, and i: link that contains the *mdp*

4.2.1.3 End Effector Target

This is the *goal-driven task* in my simulation of the working environment.

Task reference:
In this case the task reference is the *cartesian error* between the *tool* frame and the *end-effector* frame. Computed in the simulation using the *transformation buffer*, starting from names of the *end effector* and *tool* frame the translation error:

$$err^{L}_{goal/tool} = {}^{kb}(P_g - P_t) \tag{4.18}$$

all projected in the end effector frame. For the angular error I use the quaternion expression of the two rotation matrices:

$$\boldsymbol{R}^{kb}_{goal} = \rho_{goal} \quad ; \quad \boldsymbol{R}^{kb}_{tool} = \rho_{tool} \tag{4.19}$$

I than compute the error using the quaternions that gives:

$$\rho_{err} = \rho^{-1}_{goal} \cdot \rho_{tool} \tag{4.20}$$

lastly this I took the vector part of this quaternion to multiply to the rotation $\boldsymbol{R}^{kb}_{tool}$:

$$\rho_{err} = \begin{pmatrix} x_e \\ y_e \\ z_e \\ w_e \end{pmatrix} = \begin{pmatrix} \boldsymbol{v}_e \\ w_e \end{pmatrix} \quad ; \quad \boldsymbol{v}_e = \boldsymbol{u} \sin\left(\frac{\theta}{2}\right) \tag{4.21}$$

and $\boldsymbol{v}_e$ is proportional to the angle error, for small angle. The vector I used as *angular part* of the cartesian error is:

$$err^{A}_{goal/tool} = -\boldsymbol{R}^{kb}_{tool}\boldsymbol{v}_e \tag{4.22}$$

Activation function:
Since this in an *equality task* it has to be always active. And given the dimension of its *reference rate*: $\dot{\tilde{x}} \in \mathbb{R}^{6\times 1}$ the activation function is a identity matrix.

$$\boldsymbol{A} = \boldsymbol{I} \in \mathbb{R}^{6\times 6} \tag{4.23}$$

Task Jacobian:
The task jacobian is the *complete geometric jacobian* for the *end effector*, computed as shown in 4.15.

Chapter 5

Experiments

5.1

Chapter 6

Conclusions

Write the conclusions here...

Appendix A

Extra

Write here...

References

Badrloo, S., Varshosaz, M., Pirasteh, S. & Li, J. (2022). Image-based obstacle detection methods for the safe navigation of unmanned vehicles: A review. *Remote Sensing*, **14**. 3

Crespi, A., Badertscher, A., Guignard, A. & Ijspeert, A.J. (2005). Amphibot i: an amphibious snake-like robot. *Robotics and Autonomous Systems*, **50**, 163–175. vii, 10

Cursi, F., Bai, W., Yeatman, E. & Kormushev, P. (2022). Optimization of surgical robotic instrument mounting in a macro–micro manipulator setup for improving task execution. *IEEE Transactions on Robotics*, **38**, 1–17. 13

Hirose, S. & Yamada, H. (2009). Snake-like robots [tutorial]. *IEEE Robotics & Automation Magazine*, **16**, 88–98. 10

Huh, K., Park, J., Hwang, J. & Hong, D. (2008). A stereo vision-based obstacle detection system in vehicles. *Optics and Lasers in engineering*, **46**, 168–178. 4

Hussmann, S., Ringbeck, T. & Hagebeuker, B. (2008). A performance review of 3d tof vision systems in comparison to stereo vision systems. *Stereo vision*, **372**. 6

Hutchinson, S., Hager, G.D. & Corke, P.I. (2002). A tutorial on visual servo control. *IEEE transactions on robotics and automation*, **12**, 651–670. 9

Le Boudec, B., Saad, M. & Nerguizian, V. (2006). Modeling and adaptive control of redundant robots. *Mathematics and Computers in Simulation*, **71**, 395–403. 10

Maciejewski, A.A. & Klein, C.A. (1985). Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *The international journal of robotics research*, **4**, 109–117. 7, 27

Maciejewski AA, K.C. (1985). Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *The International Journal of Robotics Research*, 109–117. 10

Muslikhin, Horng, J.R., Yang, S.Y. & Wang, M.S. (2020). Object localization and depth estimation for eye-in-hand manipulator using mono camera. *IEEE Access*, **8**, 121765–121779. 4

Panasiuk, J. (2025). Controlling an industrial robot using stereo 3d vision systems with ai elements. *Sensors*, **25**. 5

Sharon, A. & Hardt, D. (1984). Enhancement of robot accuracy using endpoint feedback and a macro-micro manipulator system. In *1984 American control conference*, 1836–1845, IEEE. 12

Simetti, E. & Casalino, G. (2016). A novel practical technique to integrate inequality control objectives and task transitions in priority based control. *Journal of Intelligent & Robotic Systems*, **84**, 877–902. 8, 25

Slotine, S.B. & Siciliano, B. (1991). A general framework for managing multiple tasks in highly redundant robotic systems. In *proceeding of 5th International Conference on Advanced Robotics*, vol. 2, 1211–1216. 8

Smith, C., Karayiannidis, Y., Nalpantidis, L., Gratal, X., Qi, P., Dimarogonas, D.V. & Kragic, D. (2012). Dual arm manipulation—a survey. *Robotics and Autonomous systems*, **60**, 1340–1353. 9

Weidemann, C., Mandischer, N., van Kerkom, F., Corves, B., Hüsing, M., Kraus, T. & Garus, C. (2023). Literature review on recent trends and perspectives of collaborative robotics in work 4.0. *Robotics*, **12**. 1

Zauner, K., Dib, J.E., Gattringer, H. & Mueller, A. (2025). Workspace registration and collision detection for industrial robotics applications. *arXiv preprint arXiv:2510.23227*. 6

Zhang, W. & Sobh, T.M. (2003). Obstacle avoidance for manipulators. *Systems Analysis Modelling Simulation*, **43**, 67–74. 7

Zhou, Y., Chen, C.Y., Yang, G. & Li, Y. (2022). A sampling-based motion assignment strategy with multi-performance optimization for macro-micro robotic system. *IEEE Robotics and Automation Letters*, **7**, 11649–11656. 12

Zlajpah, L. (1997). Control of redundant robots in presence of external forces. In *Proceedings of IEEE International Conference on Intelligent Engineering Systems*, 95–100, IEEE. 11