Example of PhD Thesis with RoboticsLaTeX template

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

List of Figures

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 present on the topics of: *obstacle perception* and *avoidance* since is one of the focus of my thesis, and a key aspect of Human-Robot interaction.
And also on the topic of *High redundancy robot*, to explore possible way to deal with a structure link the one of:**SestoSenso project**.

2.1 Environment perception and awareness

Environment perception is one of the biggest different when we move from the classical use of robotic systems in the industry, to a more modern framework geared towards HRI.

2.1.1 Camera based methods

As reported in Badrloo *et al.* (2022), we can divide the camera 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 scheme:

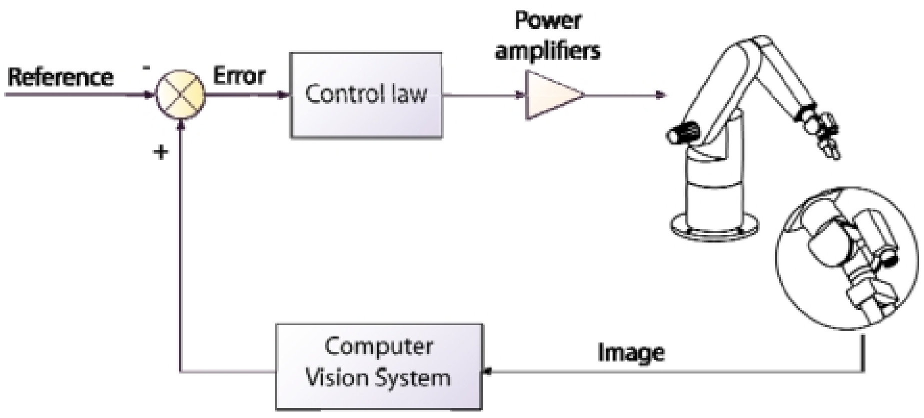

Figure 2.1: Direct visual servoing scheme

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. 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.

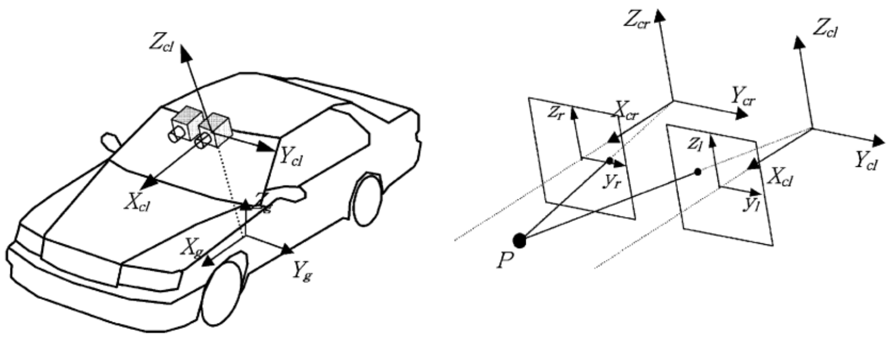

Figure 2.2: Vision coordinate sistem

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. 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**.

- **Privacy**: since the robot could work in close proximity with humans the problem of privacy of the workers has been taken into consideration.
- **Field of view**: The camera does not allow for 360 of perception around the robot.
- **Environment independence**: Camera are susceptible to change in the environment lighting, and need to be re-calibrated if the robot changes location.
- **Computational cost**: The computational cost of image recognition for dynamic environment is generally high, which increases the requirements of the systems.

2.1.2 Sensor 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.
Aside from the quality of the sensor itself, the resulting disparities between them is the amount of processing to be done on the point cloud to extract features of the environment. But they are viable to overcome the problems found with camera based methods, mentioned in the previous section.

2.2 Obstacle avoidance

Sensing the environment, especially in the case of HRC, is fundamental to do *obstacle avoidance*. 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.

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.

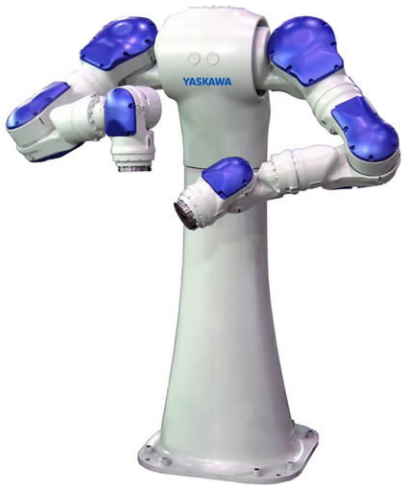

Figure 2.3: Dual-arm industrial robot example, SDA10

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.

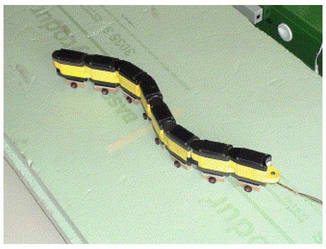

Figure 2.4: snake like robot from Crespi *et al.* (2005)

- *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. 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.

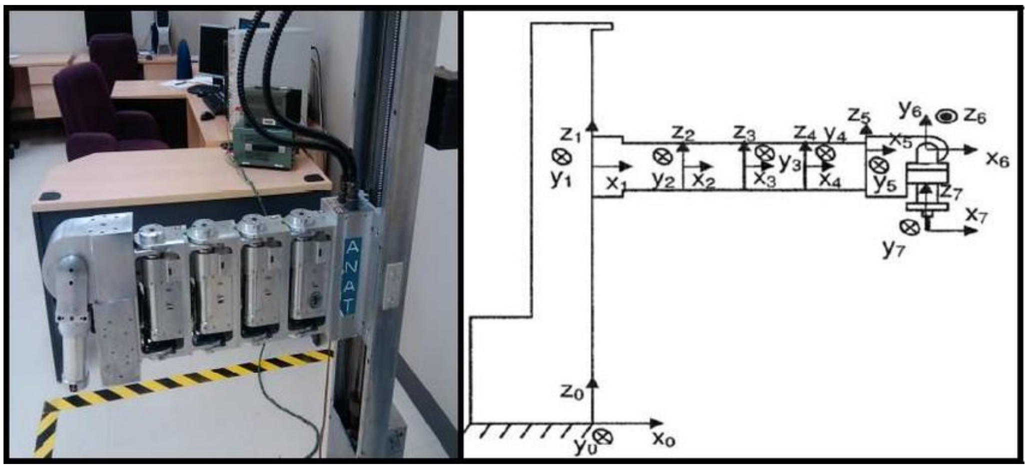

Figure 2.5: ANAT robot arm

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:

$$V_1(q) = \sum_{i=1}^{m} \sum_{j=2}^{n} \frac{\alpha_{ij}}{-(\frac{x_j - x_{ci}}{r_i + r_{si}})^2 - (\frac{y_j - y_{ci}}{r_i + r_{si}})^2 - (\frac{z_j}{h_i + h_{si}})^2 + 1} \tag{2.1}$$

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.

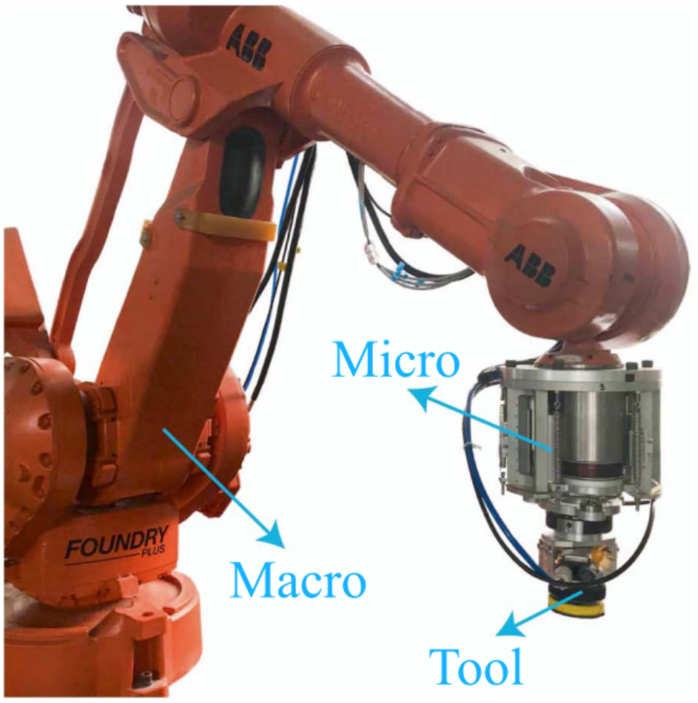

Figure 2.6: Macro-Micro robot

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).

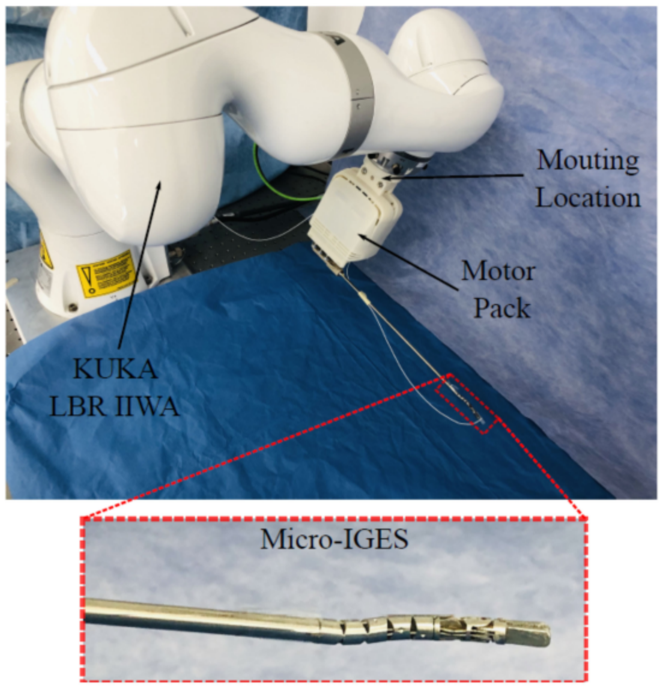

Figure 2.7: Macro-Micro surgical robot

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*.

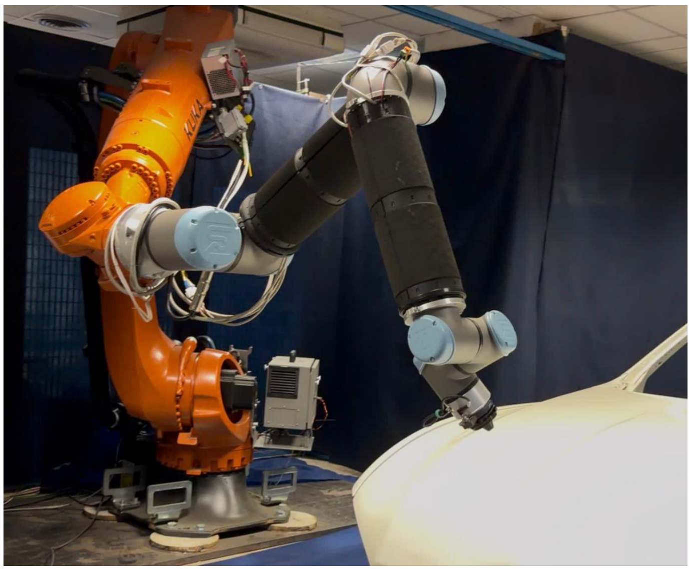

Figure 3.1: photo of the real system, inside the workcell

I started the from the work done by the team at MACLAB, and since their code already included the simulation part, I opted to incorporate their work in my architecture.

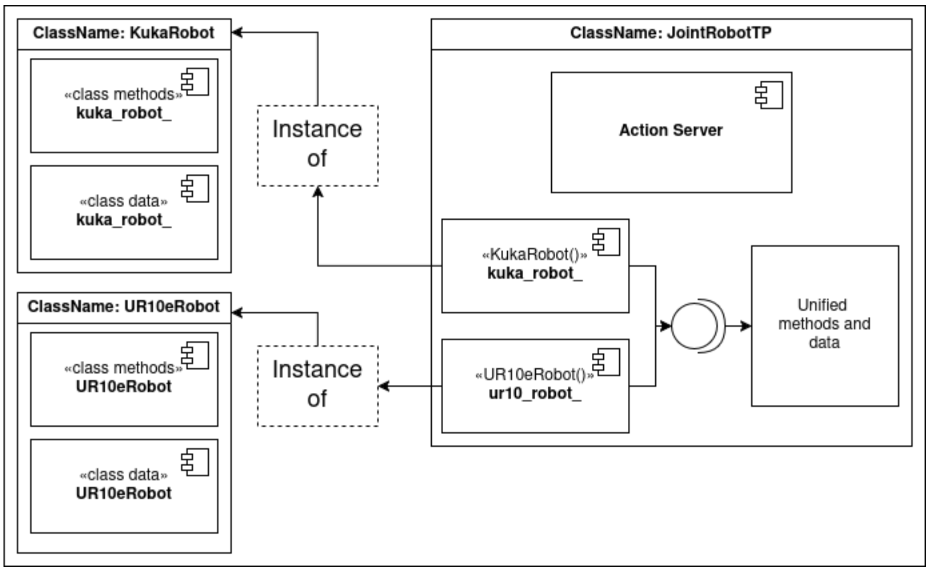

Figure 3.2: Simplified structure of the class JointRobotTP

The main objective was to use the existing classes developed for the single robots, and incorporate them in the unified architecture. The goals I wanted to pursue were the following:

1. Have an efficient way to send commands to the unified robot.
2. Use data structure that allowed me to add and remove tasks and configuration easly.
3. Keep a degree of separation between the robot representation 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 main part of the implementation of the class, are:

1. custom Action Server **RobotMoveTP**, implemented in the `uc1_robot_controllers_interfaces` package.
2. The two data structure used for the initial configuration reaching `initial_configurations_map_` and `TP_task_map_` to compute and store the matrix relative to each tasks.

3. The class used to compute each "*step*" of the task priority algorithm.

3.2.1 Action server

The custom message definition for the Action server I used to send goals to the robot is as follows:
(`<pkg> : uc1_robot_controllers_interfaces`)

```
<pkg>/MoveRobotGoal         goal
        string        init_config_name
        ------              ------
        string              result
        ------              ------
        float64           linvel_norm
        float64           angvel_norm
```

and the custom message defined for the action goal is:

```
<pkg>/MoveRobotPoint    translation
                        float64    x
                        float64    y
                        float64    z
<pkg>/MoveRobotOrient   orientation
                        float64   roll
                        float64  pitch
                        float64   yaw
```

The goal is sent from the user as a *translation* and *rotation* relative to the initial position of the end-effector(in this case I am referring to the end-effector of the *UR10e* which is the end-effector of the unified robot). The action server than uses the information from the two robots to broadcast the *Goal* with respect to the *Kuka base link*. For each *Goal* recived by the robot, I can set a different *Initial configuration*. The field called `init\_config\_name` uses a map defined inside the class to set the robot in a specific configuration before starting the *Reaching loop*.

```
std::map<std::string, Eigen::VectorXd>   init_config_name
```

By using this data structure I can use the field of the `goal` message to directly select the desired initial configuration. The set of initial configuration is defined to have interesting starting position of the robot, to analyze different behavior of the robot in the experiment part.

3.2.1.1 Action Server process flow

Here is a flow chart to better explain the functionalities of the action server implemented in the *JointRobotTP* class:

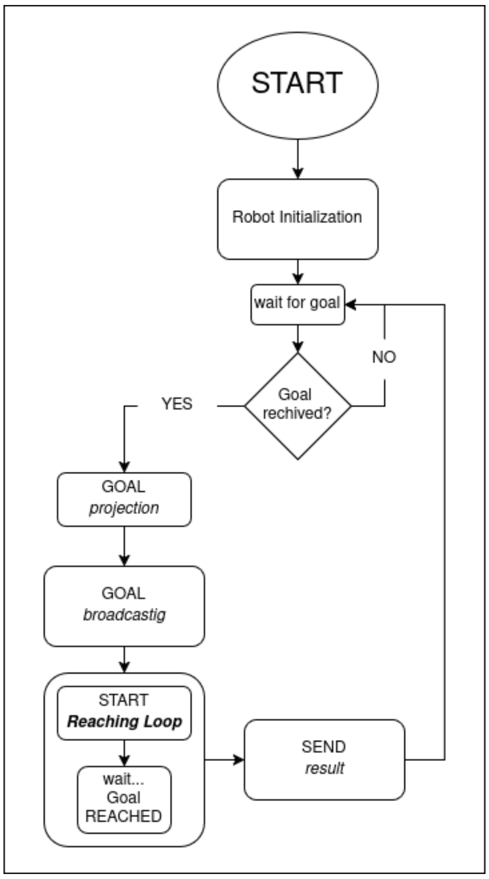

Figure 3.3: Action server flow chart

The method handling this loop is called:
`void execute(const std::shared_ptr<MoveRobotTP> goal_handle);`,
after reciveing the goal and accepting it, the requested goal is projected in the *Kuka robot base* and than broadcasted in the simulation. Lastly the method call for the *Reaching loop*, which I will explain in the next section.

3.2.2 Task Priority implementation

The control part is composed of two separate part: the data part, which is a member of the *JointRobotTP* class as for 3.2. This part is composed of a map `TP_task_map_` that contains all the matrixes relative to each task. A set of three function that are used to update the information inside the *Reaching loop*.

The definition of the data structure:
```
std::map<std::string, tp_task> TP_task_map_;
```
and `tp_task` is a `struct` with the following fields:

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

Secondly the set of functions for updating the information in `TP_task_map_` for each task have the structure:

Type	**Name**	**Args**
`void`	`Update_TRR_<task_name>`	`void`
`void`	`Update_AFunc_<task_name>`	`void`
`void`	`Update_TskJac_<task_name>`	`void`

The *Task Priority* control part is implemented trough a separate class. This class, called: `TPComputation`, has as private members two matrixes,

```
Eigen::MatrixXd    Q;
Eigen::MatrixXd  ydot;
```

these matrixes are the **projector** and the $\dot{y}$ of the last computed "*step*". Also in the initializaiton step i can define the values for the constants used in the computation of the pseudo-inverse matrix, these values will be described in 4.
As public members this class has methods to call for computing the *Task Priority algorithm*, task by task. The priority is imposed by the calling order in the *Reaching loop* code. These methods are structured as follows:

Type	Name	Args
`void`	`init_TPComputation`	`int Ndof,` `float lambda,` `float weigth,` `float treshold`
`void`	`computeTP_step`	`Eigen::MatrixXd ActFunct,` `Eigen::MatrixXd TskJacobian,` `Eigen::MatrixXd RefRate`
`Eigen::MatrixXd`	`getTP_ydot`	`void`

3.2.2.1 Reaching loop process flow

Finally I include a flow chart to inform about the process behind the *Reaching loop* implementation, for reference to the entire architecture 3.3.

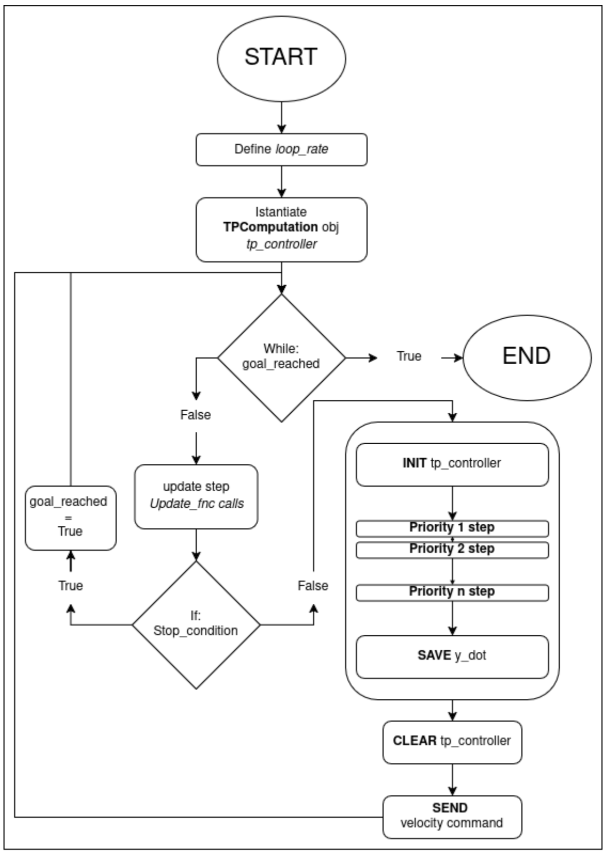

Figure 3.4: Reaching loop flow chart

The method used to implement this loop is one of the public member of the *JointRobotTP* class, namely:

```
void RunCartesianReachingLoop(std::string goal_frame, bool
reached_goal)
```

In the *Update step*, all the function created 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. Next the *tp_controller* is initialized, the variables for the pseudo-inverse are initialized in this step. Than each ”*step*” of the algorithm is computed, 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 condition is met.

Chapter 4

Methodology

4.1 Reaching Loop Description

4.2 Goal broadcasting

4.3 Task Priority

4.3.1 Task Description

4.3.1.1 Joint Limits

4.3.1.2 Obstacle Avoidance

4.3.1.3 End Effector Target

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, 7

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. 10

Hirose, S. & Yamada, H. (2009). Snake-like robots [tutorial]. *IEEE Robotics & Automation Magazine*, **16**, 88–98. 7

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

Hutchinson, S., Hager, G.D. & Corke, P.I. (2002). A tutorial on visual servo control. *IEEE transactions on robotics and automation*, **12**, 651–670. 7

Le Boudec, B., Saad, M. & Nerguizian, V. (2006). Modeling and adaptive control of redundant robots. *Mathematics and Computers in Simulation*, **71**, 395–403. 7, 8

Maciejewski AA, K.C. (1985). Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *The International Journal of Robotics Research*, 109–117. 7

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

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. 9

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. 6

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*. 5

ZHANG, W. & SOBH, T.M. (2003). Obstacle avoidance for manipulators. *Systems Analysis Modelling Simulation*, **43**, 67–74. 6

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. 9

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. 8