Investigation and implementation of adaptive workplace restrictions for the KUKA LBR iiwa

Master Thesis

Antonio Aranzana Sánchez

Aalborg University
Department of Electronic Systems

AALBORG UNIVERSITET

Here you can write something about which tools and software you have used for typesetting the document, running simulations and creating figures. If you do not know what to write, either leave this page blank or have a look at the colophon in some of your books.

Electronics and IT
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Investigation and implementation of adaptive workplace restrictions for the KUKA LBR iiwa

Theme:
Control and vision

Project Period:
Spring Semester 2022

Project Group:
1067d

Author:
Antonio Aranzana Sánchez

Supervisors:
Dimitris Chrysostomou
Sebastian Hjorth

Page Numbers: 51

Date of Completion:
June 2, 2022

Abstract:

This project is aimed on providing a adaptive solution for an already working 7 degree of freedom redundant serial manipulator controller proposed in the previous work of a student of AAU in [1]. The industrial manipulator is the KUKA LBR iiwa, for which the solution has been tested out in the actual manipulator in the laboratory. The robot's kinematics and dynamics are modeled based on the screw theory [2] and controlled by an energy-aware impedance control [3]. This project presents an advanced architecture for the implementation of virtual limitations in its controller. The new procedure is based on the analysis of entire 3D models with the use of a robust multi-plane segmentation algorithm [4]. Those planes are then inserted into the software to create the relevant constraints. The current project has proven the possibility of restricting entire areas to the manipulator successfully.

Acknowledgements

I would like to take this opportunity to give special thanks to my thesis supervisor Dimitris Chrysostomou, who's an associate professor at Aalborg University and has developed my interest into robotics manipulation and control during my studies in the master. He has also helped me developing and writing the current project along with Sebastian Hjorth, a PhD fellow at AAU I also thank enormously as he has taught me the principles of its control implementation strategy on the manipulator and helped me whenever I had a question regarding his implementation, any bug that I encountered on the way and with my own application.

I would also like to thank the closest ones, specially my parents, who have given me the opportunity to not only study abroad but supported me in every decision I made. Also, I thank my girlfriend, who's been side by side in every up and down during the entire master. I would like to acknowledge my sister, the beautiful people I had met in Denmark and my friends in Spain, who have never forgotten me no matter how less did we see each other.

Thank you.

Abstract

This project is aimed on providing a adaptive solution for an already working 7 degree of freedom redundant serial manipulator controller proposed in the previous work of a student of AAU in [1]. The industrial manipulator is the KUKA LBR iiwa, for which the solution has been tested out in the actual manipulator in the laboratory. The robot's kinematics and dynamics are modeled based on the screw theory [2] and controlled by an energy-aware impedance control [3]. Similarly to the suggested approach on Cartesian restrictions revised in [1], this projects presents a different architecture for the virtual limitations in its software. This new procedure is based on the analysis of entire 3D models with the use of a robust multi-plane segmentation algorithm [4] which extracts the planes of a point cloud on different scales depending of the application. In the end, those planes are inserted into the software to create the relevant constraints. As it was already proved to be possible to restrict the Cartesian and joint workspace of such a serial manipulator, the current project goes a step further into the implementation of entire entities restricting the workspace.

Preface

This report was written in the spring of 2022 by Antonio Aranzana Sánchez, and it describes his Master Thesis on behalf of the Electronics and IT study department at Aalborg University. The project refers to the study and implementation of an adaptive solution for the integration of virtual restrictions into the controller of an industrial manipulator. In order to understand the work presented, knowledge in Linear Algebra, Calculus, manipulator modeling, Control and Computer Vision is required.

In the thesis, the KUKA LBR iiwa robotic manipulator has been modeled and its impedance control tested on the real manipulator to its use, everything coded in C++ language with the use of the Fast-Research-Interface library from KUKA. The KUKA LBR iiwa specific data such as the Inertia Tensor, link masses, center of mass, has been extracted from the URDF provided in a free available repository found in `https://github.com/kuka-isir/iiwa_description`. The 3D models that serve as constraint entities in the controller were created as meshes in CloudCompare and its point cloud was generated in the same software with a density of 1 million points. A multi-plane detection algorithm [4] coded in C++ was used and various scripts in python and C++ were created on top for the combination of both the controller and the planes extracted from the model.

Antonio Aranzana Sánchez Aalborg University, June 1, 2022

Nomenclature

$\overline{q}_i/\underline{q}_i$	Joint bounds applied to each of the joints of the manipulator
τ^T	Total torque values calculated from the joints and wrenches
H_i^0	Homogeneous transformation matrix of the i^{th} link based on the 0 reference frame
ξ_b	Twists displacements of a body represented as a column vector
$Ad_{H_b^f}$	Adjoint homogeneous transformation matrix between the body and the inertial reference frame
$p(t)$	Trajectory of a link of a serial manipulator
p_{fb}	Coordinates of a point relative to the inertial reference frame
$q_{i,J}$	Threshold distance at which the joint limitations activates
q_i	Joint position $\in \mathbb{R}$
R_{fb}	Rotation matrix of a body link relative to the fixed reference frame
S_b	Body link reference frame
S_f	Inertial/fixed reference frame
$T(x)$	Transformation matrix from R^n to R^m dependent of x

W_C^{b,b^T} Wrench that is to be applied to repel a specific link from a constraint

Table of Contents

Chapter 1

Introduction

The industry of industrial manipulators has grown over the years as they enable performing various tasks that may be unsuitable for humans workers or harder for other robotics solution. These type robotic applications can easily reduce the labor costs - in comparison to a human worker, increment the work-safety - as it decreases the risk of a person to suffer from injury, maximize the speed of manufacturing or reduce the cycle time [5] among other pros.

It is a common believe that the robots are meant to take the work away from laborers in the production line, but this is far from reality. Indeed, robotic manipulators applied in the industry try maximize the production without replacing the workers in the production chain. Rather than that, they have co-existed for years; this mechanical units that share a workplace environment and develop their tasks along with human workers have been named Collaborative robots (co-bots). This type of robots are one of the most growing applications of the automatic applications in the industry [6]. However, they are also one of the most restrictive applications as for what has to be with the requirements to fulfill, as it can be seen in ANSI/RIA R15.06-1999 (R2009) or ISO 10218-1-2007 as examples.

Collaborative robots need to be as safe or safer than any other process in the production line. Therefore, it has always been looked for the most suitable station for the robotic manipulator. Normally, these manipulators count on fences and walls that limit the environment surrounding and thus, the reachable space of the robot arm. For the current report, the manipulator of study will be the KUKA LBR iiwa seen in Figure 1.1, a 7 degrees of freedom manipulator used in collaborative scenarios and difficult

assembly tasks.

1.1 Problem Analysis

The capacity of the industrial manipulators of behaving in a specific way taking into account the electronics, mechanical aspects and the physics applied on themselves and their surroundings is given by the control strategy implemented. Due to the nature of the robotic manipulators of being able of reaching, commonly, various meters in a 360º circumference around their base, a proper control scheme capable of reacting to the continuous changing environment is crucial.

Fig. 1.1: KUKA LBR iiwa [1]. Robotic manipulator of study of the current project.

In the literature, one of the most studied applications for the control strategies in manipulators are the reactive control designs, that can affect the behaviour of the robotic manipulator when it encounters an object in its trajectory or if any unwanted interaction takes place [7], [8]. Essentially, they react to actual collisions and stimulus in real-time. This strategies combined with the correct system design may enable the serial manipulator be aware of its are of reach and react accordingly to the unpredictable environment.

A good example of application for the combination of a reactive control such as an impedance control [9], with the addition of predefined restrictions in space to avoid known collisions has been studied previously, and would be the base and project of

[1] https://www.kuka.com/en-de

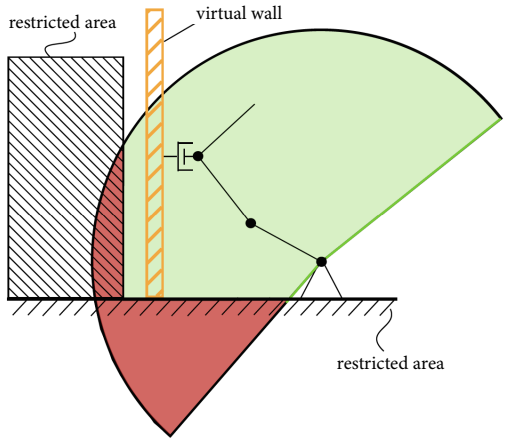


Fig. 1.2: Use case scenario for the virtual limitation in the manipulator's workspace. Extracted from [1].

reference for the current study [1]. The study emphasises en on the idea of restricting the Cartesian workspace so that the manipulator can be established in an already defined work station, making the mechanical unit adapt to its environment without the need of modifying the machinery nor the conditions of its adjacent space. With this idea in mind, this report focuses on the study and implementation of an adaptive solution for such an application, resolving on the side effect of the application demanding a study of the environment and its actual implementation on the control software. At the same time, a draft algorithm for the implementation of curved constraints is proposed, improving the overall robustness of the system.

1.2 State of the Art

As mentioned above, collaborative robots play a huge role in the industry. Therefore, along with this increase of use of such a unit, many studies dig deep into the research of minimising the collision-effect of the manipulator against the environment [7], or different solutions such as avoiding the obstacle that the robot arm would collide with [10] for a given trajectory.

These researches have in common planning trajectories and resolving on the different stated collision issues on predefined space. However, they are not suitable for real-time manipulation of the robotic arm, either programmatically or by teaching. For instance, if a worker would need to manipulate the unit to a desired position, the robot would only know how to avoid a specific obstacle blocking the trajectory the robot was programmed

to do, and thus, the new trajectory would need to be specified along with the obstacles on it in the proposed space. For any other application, the robot would fail into not colliding to a determined space.

Specifically, the control of the robotic manipulators happens in most cases as a single-input-single-output (SISO) systems, and for the handling of multi-input-multi-output (MIMO) systems such as robots, complex neural networks and fuzzy logic approximators are to be used. For instance, an interesting work presented in [11] addresses a solution for manipulator with unknown dynamics or motion constraints. But neural networks and other machine learning approaches do not normally apply alone and present themselves in combination with other schemes, such as Model-Predictive-Control (MPC).

Some studies rely on ultrasound sensors, force resistors, cameras or other solutions such that the application becomes context-aware and in advance, is able to behave according to the changing environment. For example, in [12] an MPC solution is also used utilising the response time of a worker when using a robot interface along with the time it took for him to complete the task or the number of clicks done in the interface, then controlling the system based on the external studied interactions in between the application and the human worker.

Concerning its application on obstacle-avoidance or control limitations specifically, the work of Shane et al. [13] also based on Model Predictive Control demonstrate on the ability of collision avoidance for simple motions while taking into account for specific known parameters of the obstacles, in which case are spheres and thus, they analyse the cost function of the MPC for the balls radius. This study though is rather specific for the type of obstacle analysed.

Others such as [14], [15], [16] combine context-aware strategies with the use of cameras and depth sensors for the study of Human Robot Interaction (HRI) safety avoidance techniques, adapting the manipulators trajectory based on sensors inputs. On the same path, [17], [18] limit the robotic arm planned or guided movements with the implementation of repulsive force fields and admittance control strategies respectively, achieving high-end solutions to maintain the robotic manipulator away from restricted areas.

While these strategies present a reasonable improvement of the control schemes in the devices, none of the studies in the literature show an efficient control system for already known restricted areas, which at the same time is inexpensive and suitable for a higher degree-of-freedom manipulator and for any trajectory or application of use at the same time. The research in [19], [20] investigates the last mentioned application while imposing both static and moving Cartesian restrictions onto the manipulators end-effector on top of the control strategy. Additionally, the work presented in [1] goes further into the investigation of the manipulators energy exchange with its environment, ensuring the

robots stability and safety.

This last control strategy [1] would be the base for the current project as it presents itself as the most powerful application of all the ones studied in literature, plus it is also the one most suitable to restrict common areas at a higher level approach.

1.3 Report structure

The current section will briefly define how is the document structured in case the reader consider skipping to a relevant section. The report has been divided into chapters, each chapter extending on different but related topics:

- Chapter 2: This chapter revises on the basis calculus concepts for the control of the KUKA LBR iiwa as well as the previous research done regarding the power-aware-impedance control used. Moreover, some concepts on plane detection for the constraints implementation are showed.

- Chapter 3: The main work of the master thesis is detailed. Two iterations were developed but only the second one explained in the chapter will be analysed further. Also, a draft algorithm for different surfaces than planes is described.

- Chapter 4: The validation of the model is conducted in Chapter 4 with the use of a 3D model and the actual robotic manipulator. The results are presented in force plots.

- Chapter 5: The conclusion of the report is presented along with the possible future works that the project may take.

The final objective of the current study is to validate the proposed adaptive solution for the implementation of Cartesian workplace restrictions on the KUKA LBR iiwa manipulator, stating the viability and robustness of the suggested system architecture.

Chapter 2

Robot modelling and control

In Chapter 1 it is mentioned the KUKA LBR iiwa is the industrial manipulator used for the current project along with the controller architecture of Sebastian [1], which is revised further in the following chapter to better understand the robots mathematical modeling, the control strategy itself and the system architecture behind the repulsive actions of the constraints while the robot collides with a virtual entity. Also, the approaches for the description of the constraints in terms of analysing the environment prior to the constraint implementation in the controller are studied.

2.1 Mathematical model

The definitions of a robot in the space reside on the kinematics and dynamics. These are algorithms transform motion between the so-called joint space and the Cartesian space. The kinematics of a serial manipulator represent geometrically the motion of a robotic arm taking into account the joints and the end-effector tool. On the other hand, the dynamics translate the motion at the end-effector based on the forces applied on the joints and vice versa. Both kinematics and dynamics can be sub-divided into forward and inverse calculations. To best introduce these two algorithms, some basic concepts group up the mathematical description of them.

2.1.1 Kinematics

Both kinematics and dynamics share common algebraic concepts on its calculations. One of the basis of those are transformation matrices. The transformation matrices are

what determine the rotational and translational definitions of the manipulator. Specifically, the transformation matrices that combines both of these concepts are the homogeneous transformation matrices. In linear algebra, a transformation matrix can be defined as a linear mapping between two vector spaces which maintain the properties of scalar multiplication and vector addition [21].

Considering the transformation matrix T from $\mathbb{R}^n$ to $\mathbb{R}^m$, x being a column vector constituted by n entries and A a mxn matrix, then

$$T(x) = \underbrace{\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \\ a_{m1} & \dots & & a_{mn} \end{bmatrix}}_{\text{A}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{\text{x}} \tag{2.1}$$

This equation 2.1 serves as the starting point for the definition of the most common transformations in linear algebra: the Euclidean transformations. These can be translations, rotations and reflections, but this report will focus on the first two.

The transformations of particles moving in the Euclidean space give the location of such particles at each instant of time, relative to an inertial Cartesian coordinate frame. Choosing three orthonormal axes $(x, y, z) \in \mathbb{R}^3$ such that each coordinate define the projection of the particle's location in space, we can define the trajectory of such a particle [2] as shown in 2.2. A collective set of particles that constitute a rigid body (a body link in the robot arm) needs to satisfy that *the original distance between particles where any motion or force is applied, must remain fixed* [2]. This way, the trajectories of the body links of the serial manipulator of study can be performed.

$$p(t) = (x(t), y(t), z(t)) \in \mathbb{R}^3 \tag{2.2}$$

To define the rotational component of the homogeneous transformation matrix one should take into account the orientation of the body relative to a fixed coordinate frame. Let S_f be the inertial frame, S_b the body frame and the coordinates of the principal axes of S_b relative to the fixed frame $\mathbf{x}_{fb}, \mathbf{y}_{fb}, \mathbf{z}_{fb} \in \mathbb{R}^3$, the rotational matrix is defined [2] as

$$R_{fb} = [\mathbf{x}_{fb} \quad \mathbf{y}_{fb} \quad \mathbf{z}_{fb}] \tag{2.3}$$

Considering a point p relative to the frame S_b with coordinates $x_b, y_b, z_b \in \mathbb{R}^3$, the coordinates of p relative to the inertial frame can be computed [2] as

$$p_{fb} = [\mathbf{x}_{fb} \quad \mathbf{y}_{fb} \quad \mathbf{z}_{fb}] \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} \tag{2.4}$$

thus, describing the mapping from $\mathbb{R}^3$ to $\mathbb{R}^3$ as a rigid body transformation, rotating the coordinates from the relative to the fixed frame. For the current project, the *screw theory* based modelling of the Kuka iiwa serial manipulator is adopted as in [1], where the body motions are calculated by means of rotations around a screw axis w and a translation along the same line.

In this theory, the homogeneous transformation matrix is used to calculate twist displacements, denoted as $\xi = (\nu, \omega) \in \mathbb{R}^6$, defined in terms of the linear $\nu \in \mathbb{R}^3$ and angular $\omega \in \mathbb{R}^3$ components of the rigid body [2]. The twists describe the instantaneous velocity of the targeted rigid body as

$$\xi = \begin{bmatrix} \nu \\ \omega \end{bmatrix} \tag{2.5}$$

where ν defines the linear velocity and ω the angular velocity of the rigid body.

As mentioned before, there exists two kinematics modelling: forward and inverse. The Forward or Direct Kinematics of a serial manipulator can be described as the geometric configuration of the end-effector given the known joint angles and its angular velocities [22]. Only the forward kinematics are of interest for the current project.

The orientation and position of the arbitrary point p_i expressed in the reference frame S_1 explained earlier in the section can be used for the description of each of the links of the serial manipulator with the use of the rule chain defined in equation 2.6.

$$\mathbf{H}_n^0(q_0, ..., q_n) = \mathbf{H}_1^0(q_0)\mathbf{H}_2^1(q_1)...\mathbf{H}_n^{n-1}(q_n) \tag{2.6}$$

Bringing the twists exponential $e^{\hat{\xi}_n^{1,n-1} q_n} \in \mathbb{R}^{4X4}$ dependent of an angle of rotation $q \in \mathbb{R}$ into the equation 2.7 it can be finally defined the homogeneous transformation for the robotic arm, where the exponential defines the transformation between S_{n-1} and S_n expressed in the inertial frame.

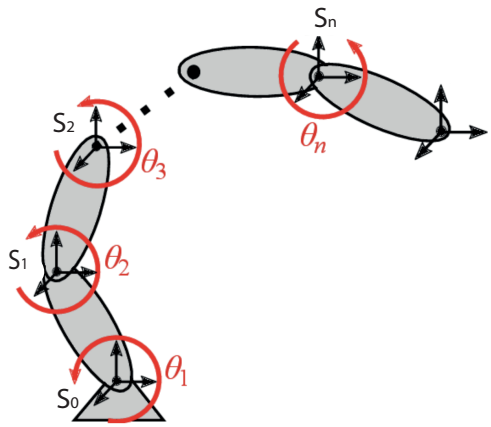


Fig. 2.1: Forward Kinematics representation. Each frame S_i representing the pose of the i^{th} link. The reference frame is denoted by S_1. Adapted from [23].

$$\mathbf{H}_n^0(q_0, ..., q_n) = e^{\hat{\xi}_1^{0,0} q_0} e^{\hat{\xi}_2^{0,1} q_1} \cdots e^{\hat{\xi}_n^{0,n-1} q_n} \mathbf{H}_n^0(0) \tag{2.7}$$

For the manipulator of study we can describe the homogeneous transformation matrix for the *tool-center-point* (tcp) or *end effector* with respect to the inertial frame as follows

$$\mathbf{H}_{tcp}^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1266 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{2.8}$$

The *tcp* is placed as indicated in figure 2.2, knowing the KUKA LBR iiwa is constituted by 7 joints, 7 links that connect those joints and a base link that actuates as the base or inertial frame and fixes the robot arm to the ground.

The adjoint of this transformation matrix between the frame S_b and S_f can also be used to express the coordinates of a twist such that

$$\xi_b^f = Ad_{\mathbf{H}_b^f} \xi_b^b \tag{2.9}$$

being $Ad_{\mathbf{H}_b^f}$ the adjoint homogeneous transformation matrix between the body and the

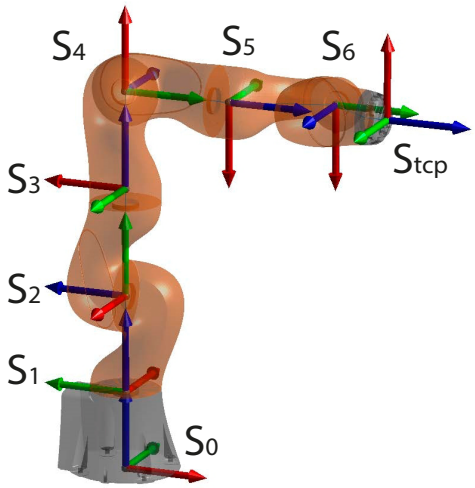


Fig. 2.2: Representation of the body frames of the KUKA LBR iiwa. Roll axis (x) being green, pitch axis (y) being red and yaw axis (z) being blue. Extracted from [5].

inertial reference frame, thus obtaining the twist ξ_b^f expressed the same way in the fixed frame S_f.

2.1.2 Dynamics

Unlike the kinematics, the dynamics of a serial manipulator describe how the robot actuates in response to the forces being applied on it, either indirectly (e.g. the own actuator opposite forces) or directly on the links. These calculations will be explained as with the basis on the *screw-theory* presented in the subsection 2.1.1.

The forces acting on a single rigid body can be explained by the combination of a force $f \in \mathbb{R}^{1x3}$ along the screw axis and a momentum $m \in \mathbb{R}^{1x3}$ [2], as seen in 2.10. This forces are defined as a *Wrench* $W \in \mathbb{R}^{1x6}$, which can also apply the theories behind the twists [2], [24].

$$W = \begin{bmatrix} f \\ m \end{bmatrix} \tag{2.10}$$

The *Poinsot's theorem* [2] allows mapping the Wrench applied onto each link of the manipulator as joint torques, in which definition the work $\mathbb{W}$ generated due to the movement of the tcp through such wrench over a time interval $t \in [t_1, t_2]$:

$$\mathbb{W} = \int_{t_1}^{t_2} WV dt \tag{2.11}$$

where the Cartesian velocity is defined by $V \in \mathbb{R}^6$ and the work applied on the wrench is assumed to be friction-less. Furthermore, if one takes the transposed Jacobian $\mathbf{J}(q)$ [25] dependent of the joint angles q the relation between the torques τ^T on the joints and the wrenches is obtained as

$$\tau^T = \mathbf{J}(q)^T W^T \tag{2.12}$$

Finally, the general equation of motion for a n-link is presented in equation 2.13 in the joint space. This equation would be derive with the force-balance-based Newton-Euler formalism described in [26] composed by a forwards and backwards iterations. This is the method being used in the current project as it is less computationally expensive than other calculus as described in [1].

$$\mathbf{M}(q)\ddot{q} + \overline{\mathbf{C}}(q, \dot{q})\dot{q} + \overline{\mathbf{G}}(q) = \tau^T \tag{2.13}$$

where $q, \dot{q}$, and $\ddot{q} \in \mathbb{R}^n$ are the positions, velocities and joint accelerations, $\mathbf{M}(q) \in \mathbb{R}^{nxn}$ express the matrix of mass, $\overline{\mathbf{C}}(q, \dot{q}) \in \mathbb{R}^n$ are the centrifugal and Coriolis torques and $\overline{\mathbf{G}}(q) \in \mathbb{R}^n$ describe the gravitational torques. This concept express the recursive inverse dynamics algorithm in form of a set of matrix equations, forming Cartesian velocities and Wrenches as stacked vectors and other block matrices that define the body twist coordinates and mass inertia [26], [27].

2.2 Reactive control

Unlike traditional robot control schemes, reactive control architectures use the current state of the robot at each time step along with its task description to calculate the joint forces applied for the very next time step. One example of this dynamics control currently used in literature for serial manipulators is the impedance control [8].

In the impedance control schemes a relation between the different state variables is made to control the manipulator, rather than controlling a single state variable as it is done in other position/velocity controllers. The impedance controller is described as a mass-spring-damper system with adjustable stiffness and damping [9], and its interaction can be seen as an energy exchange between the robotic arm and the environment [28], [29].

The description of the spring-like architecture of the Impedance controller comes in the form of a 6 dimensional vector and describes the motion generated of the end-effector from its current transformation $\mathbf{H}^0_{tcp}$ towards its goal $\mathbf{H}^0_d$, modeled as a Wrench [28] as in 2.10. In the current project, an Energy/Power-aware Impedance controller first developed in [30] is used as in [1]. Specifically, this control strategy uses both Energy shaping and Damping Injection [31].

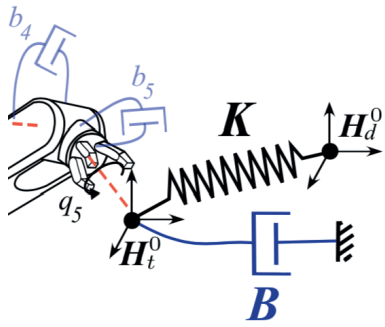


Fig. 2.3: Illustration of the spring + damper control. K representing the Cartesian spring between the tcp and the desired transformation. B represents the Cartesian damping while b_5 is the damping of the joint at the end-effector.

Definition of a spring

As introduced before in the chapter, the impedance controller is described as a mass-spring-damper system. Specifically, the spring is mathematically described as in 2.10, exactly like a Wrench 2.14, which is also the same way as any force acting on a rigid body can be modeled as [2].

$$W_K^{tcp,tcp^T} = \begin{bmatrix} f_K^{tcp,tcp^T} \\ m_K^{tcp,tcp^T} \end{bmatrix} \tag{2.14}$$

In equation 2.14, all the variables are expressed in terms of the Cartesian spring K relative to the tool-center-point of the robotic manipulator.

2.2.1 Cartesian constraints

In Hjorth's research [1] the definition of the Cartesian constraints are described as a smooth manifold $C \in \mathbb{R}^3$. Those, actuate as virtual walls in the controller in the form of 2D planes in the Cartesian workspace. To create those planes, 3 independent points

$P_1, P_2, P_3 \in \mathbb{R}^3$ are to be defined in space to be able to derive the normal vector with the points cross-product.

$$n_{j_x}x + n_{j_y}y + n_{j_z}z + \vec{n}_j \cdot P_0 = 0 \tag{2.15}$$

Moreover, once the plane is described in space as shown in 2.15, the projection of the point of study onto the plane needs to be computed.

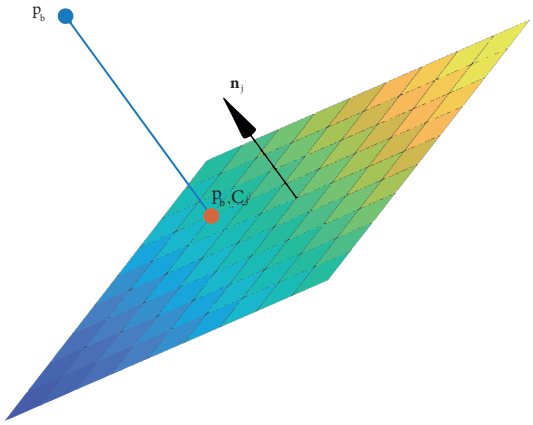


Fig. 2.4: Visualization of the projection of a point p_b into a plane along with its normal vector $\vec{n}_j$.

The projection of any point in space onto a plane can be described as in 2.16. With these values, the wrench-like definition can compute its force and momentum applied on the point of interest, which in this case it will be the end-effector, and thus, the resulting torque that may be applied onto the point contrary to its current movement to keep it away of the constraint.

$$p_b, C_j = p_b + \vec{n}_j t = \begin{bmatrix} p_{b_x} \\ p_{b_y} \\ p_{b_z} \end{bmatrix} + \begin{bmatrix} n_{j_x} \\ n_{j_y} \\ n_{j_z} \end{bmatrix} t \tag{2.16}$$

Where p_b describes the position of the b^{th} link in $\mathbb{R}^3$ in reference to the manipulator's inertial frame, C_j represents the constraint, $\vec{n}_j$ is the normal vector derived from such a constraint and t defines the distance measured between the end-effector/any other joint of the serial manipulator in respect to the constraint. The final expression for the wrench calculations in terms of the constraint application onto the end-effector of the

manipulator relative to the base reference frame used for the calculation of the τ_{motion} in equation 2.18 is

$$W_{C_j}^{b,b^T} = \begin{bmatrix} f_{C_j}^{b,b^T} \\ m_{C_j}^{b,b^T} \end{bmatrix} \tag{2.17}$$

where $W_{C_j}^{b,b^T}$ expresses the repelling wrench that is to be applied to keep the link b within the Constraint C_j, based on its force and momentum as defined in equation 2.10.

From [1] it can be extracted the resulting joint torques of the serial manipulator of study 2.18, which essentially is the combination of both the Wrench generated by the spring 2.14 and the constraint 2.17, described previously in 2.2.

$$\tau_{motion}^T = \tilde{\mathbf{J}}^T(q) \left(W_K^{0,tcp^T} - \sum_{b=1}^{n} \left(\sum_{j=1}^{m} W_{C_j}^{0,b^T} \right) \right) \tag{2.18}$$

where n is the number of joints, m is the number of constraints, W_K^{0,tcp^T} is the elastic wrench, $W_{C_j}^{0,b}$ is the wrench generated by the constraint relative to the base reference frame and $\tilde{\mathbf{J}}^T(p)$ is the spatial Jacobian describing the links spatial velocity in the same reference frame.

2.2.2 Damping injection

After having defined the spring, the damping $\mathbf{B}$ matrix is left to be described. As modeling the damping $\mathbf{B}$ constant is not ideal [1], the values of $\mathbf{B}$ will be dependent of the state of the system such that $\mathbf{B}(q) \in \mathbb{R}^{nxn}$ resulting from the multiplication of the diagonal elements of the Mass matrix $\mathbf{M}_{diag}(q)$ and $\mathbf{B}$.

$$\mathbf{B}(q) = \mathbf{M}_{diag}(q)\mathbf{B} \tag{2.19}$$

On top of the damping definition, the *damping injection* method is usually implemented [3] so that the energy exchange in between the robot and the environment P_{motion} is limited. This, is controlled by a scaling factor $\beta \in \mathbb{R}$, which will apply whenever the threshold P_{max} of the energy the robot would exchange with the environment is violated [1], [3].

$$P_{motion} = \left(\tilde{\mathbf{J}}(q) {\mathbf{W_K}}^{0,tcp^T} - \mathbf{B}_{init}(q)\dot{q} \right)^T \dot{q} \tag{2.20}$$

where $\mathbf{B}_{init}(q)$ describes the damping prior to is multiplication with the factor β, which value can be calculated from 2.21.

$$\beta = \begin{cases} 1 & \text{if} \quad P_{motion} \leq P_{max} \\ \dfrac{\left(\tilde{\mathbf{J}}(q)^T {\mathbf{W_K}}^{0,tcp^T}\right)^T \dot{q} - Pmax}{\dot{q}^T \mathbf{B}_{init}(q)\dot{q}} & \text{otherwise} \end{cases} \tag{2.21}$$

The final calculation of the damping term comes as in 2.22.

$${\tau_{damping}}^T = \beta \mathbf{B}(q)\dot{q} \tag{2.22}$$

2.2.3 Joint limit avoidance

As the robot is capable of reaching any point in space independent of the current joint configuration, it is very likely to reach a joint limit, where the robot needs to be re-initialised as it goes into a backlog. The method called *joint limit avoidance* as studied in the literature [32] is then introduced. This method is similar on the way the manipulator's links are kept within a constraint. The concept is described as a relation between the predefined maximal and minimal bounds $\overline{q}_i/\underline{q}_i$, the distance at which the constraint activates $\overline{q}_{i,J}/\underline{q}_{i,J}$ and a scaling factor $\Omega \geq 0$.

$$\tau_{q_i}(q) = \begin{cases} \frac{\Omega}{\underline{q}_i^2} \left(\frac{1}{\underline{q}_i} - \frac{1}{\underline{q}_{i,j}} \right) & \text{if} \quad \underline{q}_i \leq \underline{q}_{i,J} \\ -\frac{\Omega}{\overline{q}_i^2} \left(\frac{1}{\overline{q}_i} - \frac{1}{\overline{q}_{i,j}} \right) & \text{if} \quad \overline{q}_i \leq \overline{q}_{i,J} \\ 0 & \text{otherwise} \end{cases} \tag{2.23}$$

An illustration of this approach is showed in Figure 2.5, noticing the torque is dependent of the current joint position q_i and its difference to the upper/lower joint limits $\overline{q_i}/\underline{q_i}$.

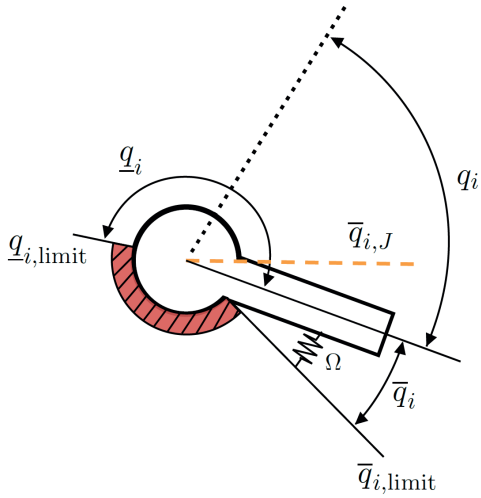


Fig. 2.5: Visualization of the joint limit avoidance concept. The red coloured area represents where the manipulator joint is not allowed to reach physically, the black dots is the line marking the midpoint of the joint and the dashed orange line represents the distance to the upper limit of the constraint.

2.3 Controller architecture

The resulting torques applied on the serial manipulator can be calculated as the sum of the torques explained earlier in this chapter. From those, only the damping torque of the spring effect described in the energy-aware impedance strategy is opposite to the rest of the calculated torques.

$$\tau^T = \tau^T_{motion} - \tau^T_{damping} + \tau^T_{CC} + \tau^T_{JLA} + \tau^T_{Coriolis} + \tau^T_{Gravity} \tag{2.24}$$

where $tau^T_{Coriolis}$ and $\tau^T_{Gravity}$ are the torques representing the compensation for the Coriolis and gravitational forces respectively, τ^T_{JLA} is the torque computed by the joint limit avoidance, τ^T_{CC} defines the torque of the Cartesian constraint, τ^T_{motion} describes the actuation of the spring and finally $\tau_{damping}$ is derived from 2.22. The overall structure of the control system involving the impedance control, the virtual constraints and the joint limitations looks like the one seen in Figure 2.6.

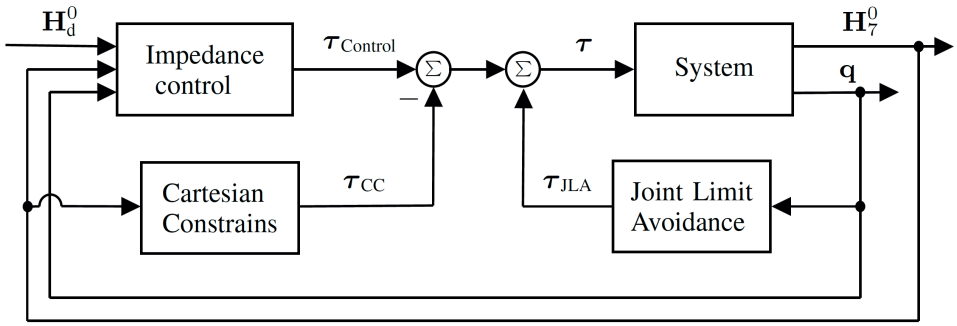


Fig. 2.6: Overall control strategy implemented. Extracted from [33].

2.4 Constraints definitions in space

As described in subsection 2.2.1, the base project defines the Cartesian constraints implemented in the controller of the manipulator as 2D planes. For describing such a planar constraint in space, at least 3 points were expected to be hard-coded in the software such that the normal vector of the constraint is computed and inputted in the wrench force and momentum calculus. To avoid hard-coding any instances in the software and produce a more efficient and robust system an implementation of constraints detection for the broadest scenario possible has been studied.

Several adaptive solutions for constraint detection were first introduced in section 1.2. One of the most suitable solutions for the current application is considered to be the detection of planes directly taken from 3D models, as it enables the software to calculate the points in space that can be inputted into the running system to impose the constraints explained in section 2.2.1. This specific application regarding the search of planes for the creation of virtual constraints inside the manipulators controller has not been combined before in literature. The utility this brings resides on implementing the serial manipulator in any other facility, which normally the companies support owning the physical modelling of the emplacement for the robotic arm, and whether this is not the case, it can be easily extracted with the use of a simple camera (e.g. from the Intel RealSense family [34]).

Taking this adaptive solution into practice, various iterations were done and detailed in Chapter 3. The final solution is achieved with the use of a plane detection algorithm that is able to analyse the constraints efficiently and thus, the software was adapted to impose the restrictions based on the findings. The different algorithms found in the literature will then be explained further in this section.

2.4.1 Plane detection algorithms

It is well known that there exists machine learning and deep learning algorithms capable of detecting planes out of images in an easy way and computationally faster than other traditional algorithms. An example of this is the work of Chen Liu et al. [35], where they use a RCNN for the computation of the different planar surfaces from RBG images. As the work is mainly focused on detecting planes on images, Zehao Yu et al. [36] combine a fast-detection method using CNN and a mean shift clustering algorithm to cluster the masked pixel embedding into plane instances, in an attempt to reconstruct the 3D environment.

Despite the fact that their accuracy on plane detection is good enough for the case of use, in the literature these approaches have only been studied for single images analysis with 2D inputs, while a 3D model is required. For such reason, some other traditional methods may fit better. Indeed, there is a concept that can describe a 3D model in space giving a position for each point of a surface given an origin: point clouds [37]. Unlike 2D images, point clouds give 3D structural data of an object. Point clouds can be recovered from an RGB-D image when a camera's intrinsic parameters are known.

Some of the traditional algorithms for planar surface segmentation using point clouds include the research done in [38], which utilises the RANSAC algorithm to determine the probability of detecting a model using the minimal set of data to estimate it. Additionally, they compute a voxel growing algorithm having estimated the normal vectors for the points in respect to the surface.

On the other hand, the much newer study for multi-scale planar segmentation [4] offers a way more robust software, as it is able to detect millimeter-scale planes in large point cloud data sets. It also offers more flexibility as the algorithm can be computed for different scale values, detecting either simple small shapes or bigger areas that share a common surface as seen in Figure 2.7. The only real downside of this specific work is the time it takes to pre-process the data in order to be able to input it to the system, however, the current work does not require real-time processing as the inspection of the 3D model is supposed to be done prior to the robotic manipulator implementation in the actual field.

Fig. 2.7: Plane extraction algorithm for the search of constraints inside a 3D model of a church for 2 different scales. Extracted from [4].

Utilising such an algorithm to the estimation of planes in a 3D model can then be used to used the points in space of a specific plane to limit the manipulators reachable space around the area. This algorithm will be the one used in the current study and explained further in Chapter 3.

Chapter 3

Model study

So far, the state of the art on previous implementation on workspace limitations along with basic concepts regarding the robot's manipulation and the control being used have been covered. In this Chapter, further detailed explanations on virtual constraints addition improving the overall efficiency and robustness of the system are described. The final proposed adaptive solution is illustrated in section 3.3.

3.1 Virtual Walls

Previously in section 2.2.1 an approach on virtual wall-like constraints is proposed. This method emphasises on the idea of the projection of the point that's about to hit against such constraint to understand the distance of activation of the repulsive force needed to repel the point. The only requirement for it is to know the position and the normal vector the constraint points to as it defines the direction of the repelling force. However, this application also provides some limitations for two main aspects: these virtual walls were hard-coded into the software, what makes the claim of 'adaptability to the environment' very vague in terms of work needed to be able to accomplish this. And secondly, only virtual planes that are infinite along an axis are allowed to be added. As suggested in [33], a more complex implementation of the constraints along with an adaptive solution for it is studied.

In the current report, two different approaches on an adaptive implementation of the virtual control limitations were implemented. The first one studies the viability of the system on adding constraints while executing the application. The second approach pays

an special attention to the robustness of the solution in terms of flexibility, however it is expected to be used prior to the manipulator's use.

3.1.1 First iteration: individual walls

The first approach tested is almost entirely software related. It does use the basic concepts explained in 2 such as the calculations of the position of the TCP with the use of the forward kinematics or the wrench-like modelling for the repulsive force into the joints. It offers a real-time wall placement of a constraint based on input points calculated from the end-effector's position.

The software-base concept that enables inputting these values while running a trajectory in the manipulator with the previous proposed controller are the threads. A thread in computer science is the smallest sequence of instructions that are to be managed independently inside a program [39]. Its implementation into the code is easy to replicate as the common libraries are already created by Microsoft and ready to be included.

Once a thread is open, the logic behind the application was as showed in the following pseudocode:

Algorithm 1 Pseudocode of the secondary thread for implementation of constraints in real-time

```
1:  procedure THREAD
2:      Npoints = 0
3:      while True do
4:          if CheckForInput = True then
5:              TCP_pos = CalculateFK()
6:              StorePoint = TCP_pos
7:              Npoints++
8:              if Npoints = 3 then
9:                  Plane = CalculatePlane(StorePoints)
10:                 SetConstraint(Plane)
11:                 Npoints = 0
12:             end if
13:         end if
14:     end while
15: end procedure
```

This application offers great flexibility on constraint implementation if it is required as a fast solution. Nonetheless, this method is still insufficient as it does not permit either implementing other type of virtual constraints than planar surfaces nor multiple constraints implementation at a time.

3.1.2 Second iteration: 3D model

It has been mentioned the search for greater robustness and adaptability of the system in terms of the different constraints shape or the number of constraints integrated into the system at a time. With this objective in mind, it makes common sense to start thinking about whole 3D model inputs into the system. This task brought new questions such as the input type that the system required of what was to be analysed.

As revised in the previous sections 2.2.1, the unchanged attribute in the virtual walls integration are the points for the plane creation, which calculation is far from laborious and that makes it specially good for its software computation. A technology very well known in literature for 3D model representation is point clouds. Point clouds are essentially a set of data points in space, and they are the prior step to the creation of accurate 3D modelling [37]. Point clouds can normally be obtained through modern lasers or cameras, but they also might be obtained digitally by online softwares such as MeshLab or CloudCompare.

In order to be able to analyse a specific model input into the software, an own 3D model was created with the use of CloudCompare.

Simulation environment

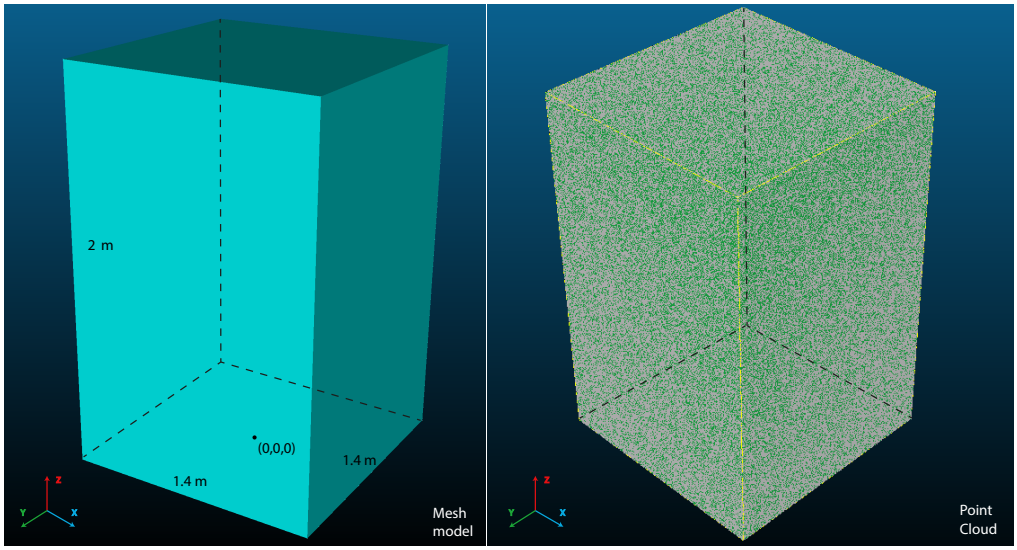


Fig. 3.1: Illustration of the 3D model that is used as an input to the system, where the constraints will be placed.

As shown in Figure 3.1, a box of dimensions 1.4x1.4x2 m was created in the preferred software. After some iterations, a point cloud of 1.000.000 points - enough to not slow down the software - from the mesh model was then computed so that it is easier to work with the model and the positions of all points in space can be obtained. Note that the less number of points chosen for the model design the fastest the software will analyse the point cloud, but the less precise it will be. At the same time, the normal vectors of the points were calculated in CloudCompare such that we can understand the direction of the surface of each point. The calculation of such normals from a the point cloud are computed with a Randomized Hough Transform that serves as an image for the discrete probability distribution of the possible normals [40].

Both software and normal calculation method are proved to be suitable for any surface model [40] and thus, either applying this exact same calculations to a point cloud obtained either from a RealSense camera or a CAD that already exists will output a similar result. Therefore, despite the current application model is based on planar surfaces, some other curved surfaces or more complex input models may be applied too.

3.2 Plane detection onto the model

Once a model input is selected and the point cloud is processed, a way to bring those points into constraints must be devised. In order to produce the simplest transition from the previous iteration to the current one, it was decided to continue loading constraints as planes from a set of 3 points. This way, the software was missing on understanding the points from the mesh as unified planes.

In the literature, it was revised in 2.4.1 the different strategies to segment point clouds into planes. Among all, the work of [4] appears to be way more robust and flexible than others, and it is the one to be studied further.

The software architecture followed in the paper includes pre-processing the data, run the plane segmentation algorithm and compute some topological analysis onto the segmented output.

3.2.1 Data processing

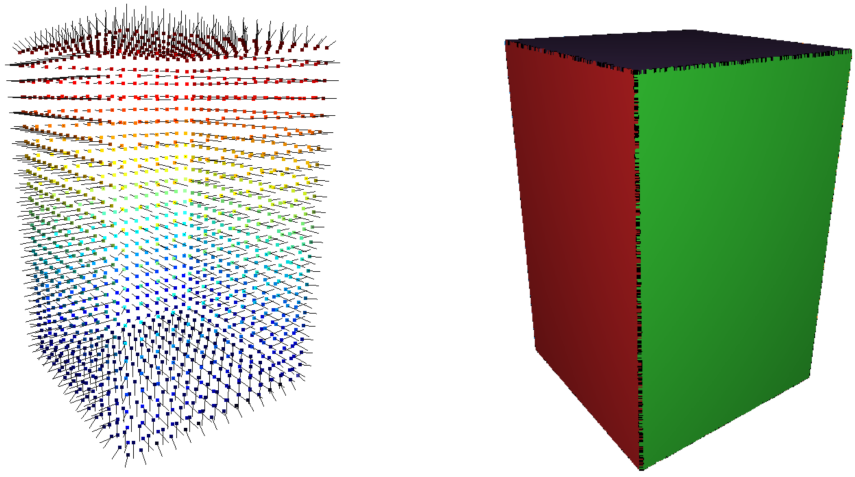
The first step is preparing the data that is going to be analysed. For doing so, they compute a set of per-scale normal vectors from local surface reconstructions for each point in the point cloud. Then, they simply group up the points at a certain scale in *regions* with the use of a region growing algorithm. This regions will serve as what they call *nodes*. The limits colliding of a region will at the same time be the *edges*. The end of the data processing ends with the calculations of the *components*, which are sets of

regions that are connected through the edges. With all these amount of data, they are able to finally compute the persistence analysis of the extracted components.

In the Figure 3.2b, the set of computed extracted normals from the Box model is showed.

3.2.2 Topological analysis

The components stored have additional values: birth and death scales that represent the levels of the regions. With these values and the Topological Data Analysis toolbox, they are capable of determine the lower level difference to be meaningful planes at lower scales while the higher level difference to be higher scales surfaces.



(a) Box normal vectors pointing towards the orthogonal direction of the surface where they are in.

(b) Different planar surfaces found in the box model. Each color represents one individual plane.

Fig. 3.2: Examples of the different stages of the software used: on the left side, pre-processed downsampled point cloud stage; on the right side, the actual output of the software at a high scale level.

In the paper [4] they also propose different thresholding methods to evaluate the point clouds depending on the persistence level (birth - death values) or the scale at what the planes will be looked for. The final solution given is proved to be robust for a wide variety of 3D complex models and was compare to common RANSAC algorithms amongst others. The results showed an impressively low Root Mean Square (RMS) error on their approach with a more than decent plane extraction. The only major downside of the application might be the time of computation, as the extraction of the components increase the time it takes to perform the entire cycle quite a lot. However, this can be

easily solved if the point cloud is first down-sampled to a decent amount. The quantity of points needed for a specific situation varies from model to model, specifically, for a robotic manipulator in an industrial environment, a set of 1 million points would be more than enough as it is expected to control larger areas and not tiny shapes with sharp edges.

3.3 Overall software structure

The final solution is presented as an adaptive constraints implementation into the control of a serial manipulator. This procedure takes several steps, as shown in Figure 3.3. First, either a 3D model that can be converted into point cloud or a point cloud achieved from a modern camera is needed. Then, the data of the model is analysed to execute the plane extractor software. Finally, from each of the computed planes, 3 points are selected so that a virtual plane can be generated inside the software, thus creating a virtual constraint in the axis of the plane extracted.

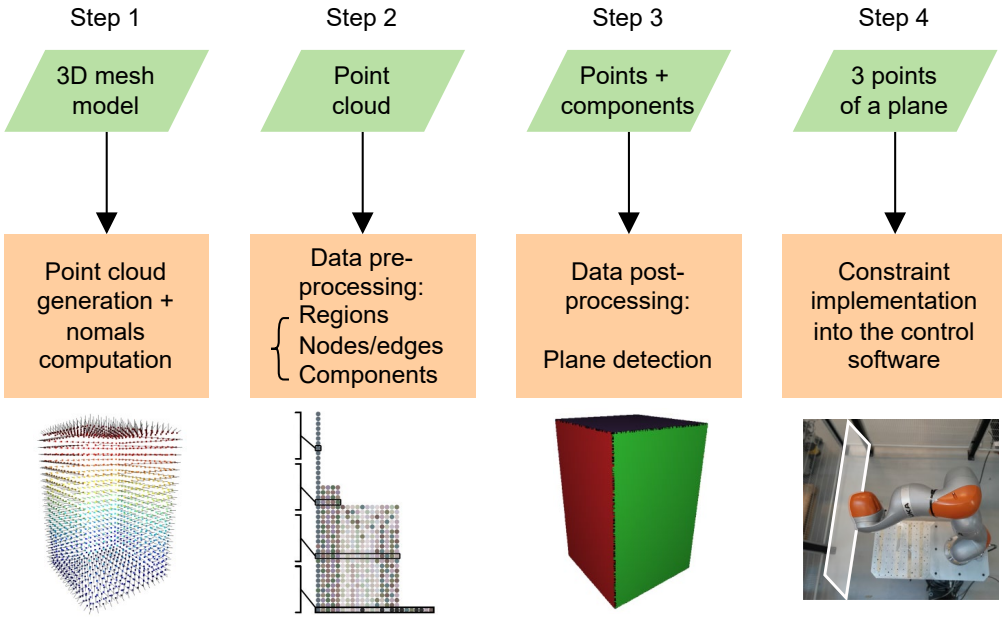


Fig. 3.3: Visualization of the overall software structure for constraints implementation from 3D model inputs.

Despite that this solution integrates entire 3D model constraints, they are only viable for planar surfaces that extend indefinitely in space. For such reason, a proposal that includes logic for the handling of multiple-planes into the controller is described in the upcoming section.

3.4 Curved constraints solution

Integrating more complex surfaces is possible when a solution for planar surfaces has already been proposed. Indeed, the idea is based on bringing the effect performed with the planes at a lower scale. As mentioned in 3.1, the plane detection focuses mainly on the normal vector computations from the model. Also, as seen in Figure 3.2b, those normal vector computations are being done for all points in the sets.

If the simplest curved model is brought to the scene, we can end up with what is illustrated in Figure 3.4. Bringing the regions concept revised in [4] to a somewhat low scale plane computation, close points share enough similarities to be scanned as unique planes depending on the scaling factor as shown in the right side of the figure.

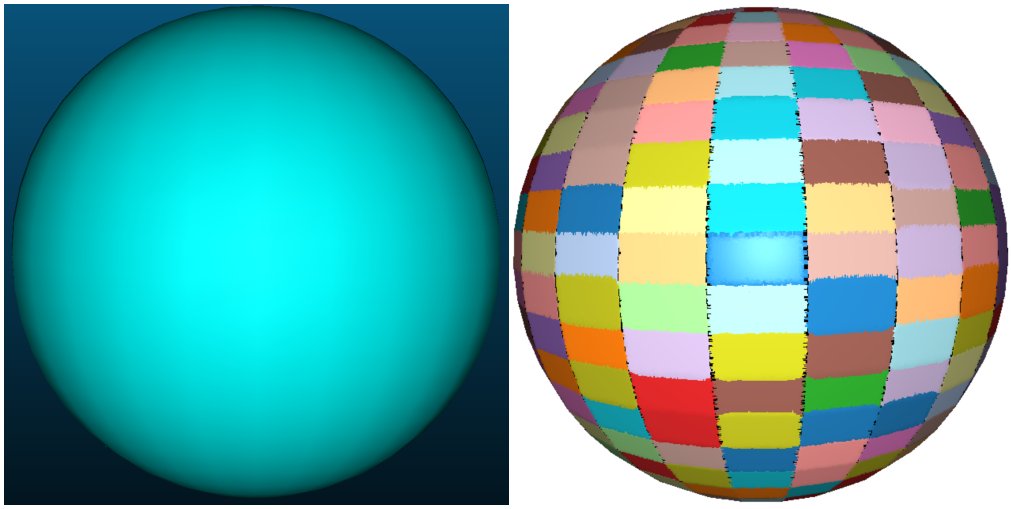


Fig. 3.4: Sphere model on the left side. On the right side, the grouped points or planes-like regions detected on the sphere.

After having processed the plane segmentation on the curved surface, a logic can be created into the software such that each of the constraints (planes) activates only whether the projection of the required repelling point actually is on such plane. This way, the

architecture would look like a constraint that's imposed or not dependent of the projection of the end-effector. This same idea might be brought to lower scales multi-plane computations to treat similar and more complex model structures.

Algorithm 2 Pseudocode of the logic for curved constraints handling.

1: $vertices_{plane} << knownvalue$
2: $TCP_{pos} = CalculateFK()$
3: $projection = calculateProjection(TCP_{pos})$
4: **if** $vertices_{plane} < projection[x, y, z] < vertices_{plane}$ **then**
5: $\quad calculate : distance, Wrench, torque$
6: **else**
7: $\quad$ // the wrench will not apply in this case
8: **end if**

The logic is self-explanatory, but essentially it will calculate the torque values for the end-effector to stay away from the constraint depending on whether the projection is within the limits of the constraint. This logic will also alleviate the software from unnecessary computations in the case where the robot needs to handle several constraints which it should not collide with at that time and so, the wrench values and torques will not be calculated.

Chapter 4

Model Validation

The upcoming Chapter will focus on presenting the scenario and the tests required to analyse whether the proposed system is reliable according to the aimed objective in Chapter 1. The final goal includes creating a software model that is capable of analyse 3D point clouds, extract the regions the manipulator will collide depending on its position and at the same time, controlling the manipulator based on those constraints imposed. Similarly, the multi-plane extraction system is independently validated in [4] as well as the control strategy presented in [33].

To better define the objective and be able to determine the outcome of the tests, a set of requirements are listed below:

The KUKA LBR iiwa must stay within the imposed constraints:
1. Never being physically able to reach the actual constraint. 2. The controller providing the robot more power as the robot approaches the constraint. 3. Stay within the limits when encountering more than 1 constraint.

As the target is validating the constraints position in space along with the force applied by these, every constraint in the input model will be analysed for its relevant axis. It is therefore expected to see an increase in the Cartesian force in the axis of the collision to the constraint. For instance, let's say we have a constraint in the plane x = 0.7, as the origin is considered to be the center of the robot, it is common sense to assume the resulting Cartesian force of the constraint into the end-effector should point towards the x-axis in the opposite direction the robot is trying to push.

4.1 Testing environment

Despite it has been studied a system architecture where more complex models with curved surfaces as shown in 3.4, the tests will be conducted only on the box-like model described along the sections 3.1, 3.2b.

It is also worth mentioning that the 3D model has a height of 2 meters. As the KUKA LBR iiwa is placed on top of a table and the software estimates planes in the entire point cloud, it is known that the bottom constraint at $z = -1$ and the robot can not hit it, therefore it has been erased and and new constraint placed at $z = 0$. Instead of doing this, a different model could have been used for the tests, which is also revised in the section 5.1.

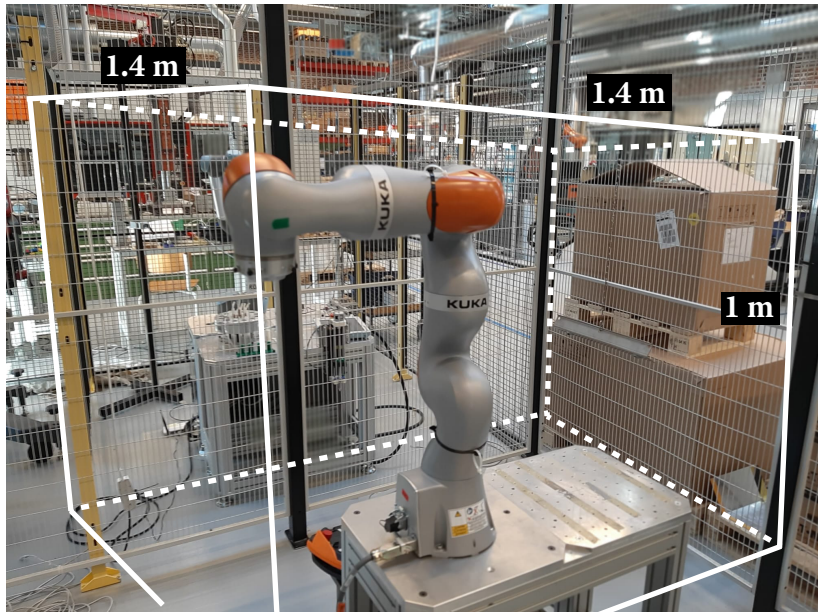


Fig. 4.1: Simulation environment: virtual box constraint inputted in the system.

In the the Figure 4.1 it can be seen the virtual constraints that are to be imposed in the controller. The fences of the laboratory surrounding the robot in the figure do not correspond to the virtual box illustrated in white, plus the dimensions of the fences are larger and so the manipulator cannot ever collide with them. Subsequently, the robot colliding with every virtual constraint imposed will be shown individually plus an additional demonstration of multi-collision in the end.

Fig. 4.2: Starting robot position with the origin being the base link and the end effector placed at (-0.4, 0, 0.6)m approximately. The roll axis (x) coloured in green, pitch axis (y) coloured in red and yaw axis (z) in blue.

4.2 Simple collision tests

For all the tests, the robot is set to be at a fixed starting position that makes the end-effector be approximately in the point $(-0.4, 0, 0.6)$.

That being said, the first test conducted was in the x axis, where it is to be expected two constraints at $x = 0.7$ and $x = -0.7$.

Plane constraint at x = 0.7

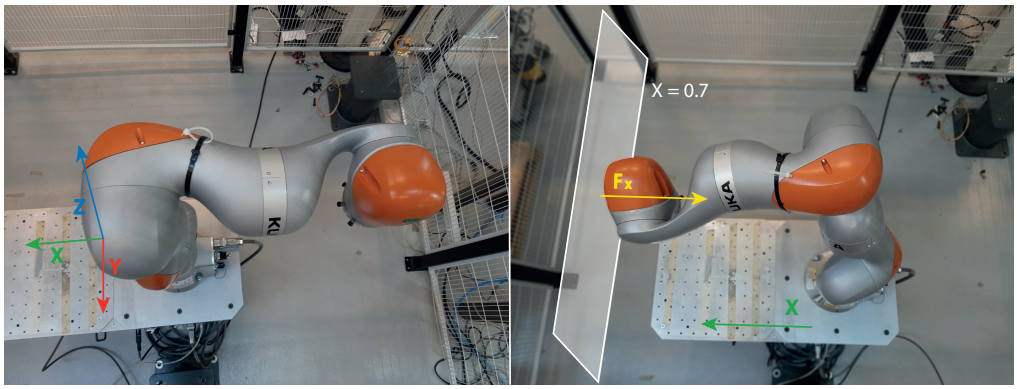


Fig. 4.3: Displacement of the robot to a position where it collides directly with the constraint of the cube imposed at x = 0.7m.

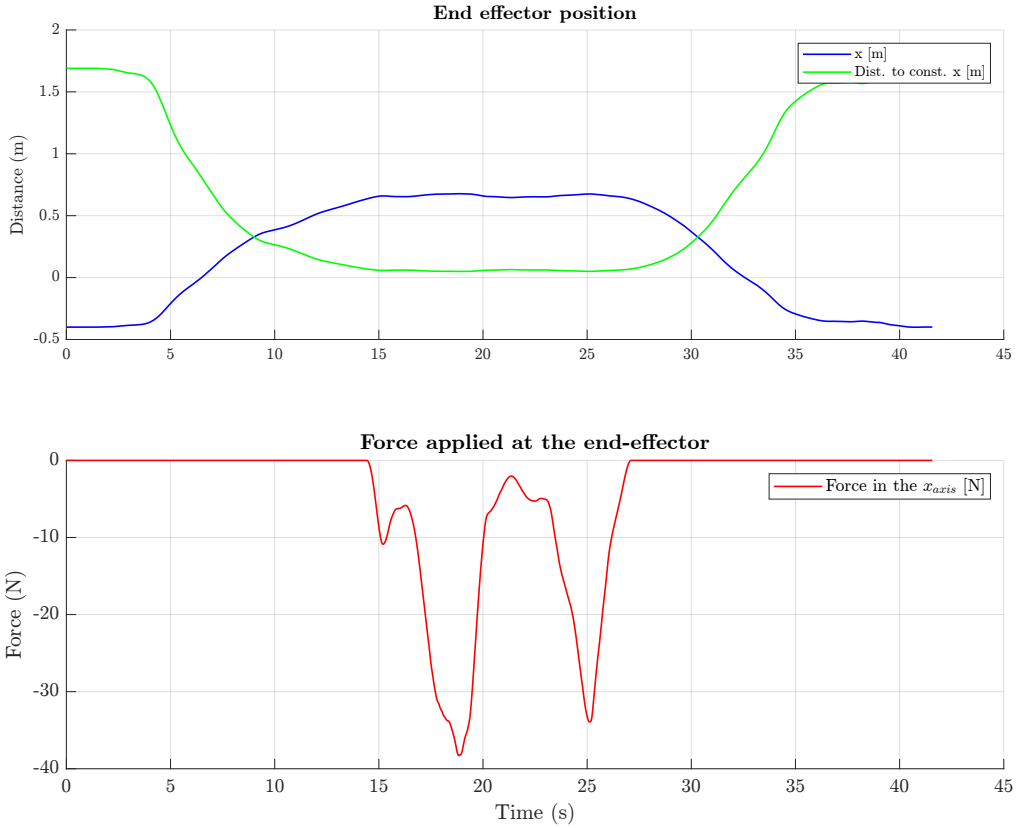


Fig. 4.4: End-effector's trajectory along the X axis in the first figure, followed by the Cartesian force applied onto the end-effector on the same trajectory.

In the Figure 4.4 it can be appreciated how the force the constraints applies on the end-effector increases as it reaches the virtual wall at x = 0.7. The force increments in the opposite direction of the movement the harder the robot is pushed against the constraint, only experiencing a non 0 value when it hits the wall. This happens at around 14 seconds into the execution of the test. After the second 27, the robot is released and stabilises itself in the initial position thanks to the impedance control.

Plane constraint at x = -0.7

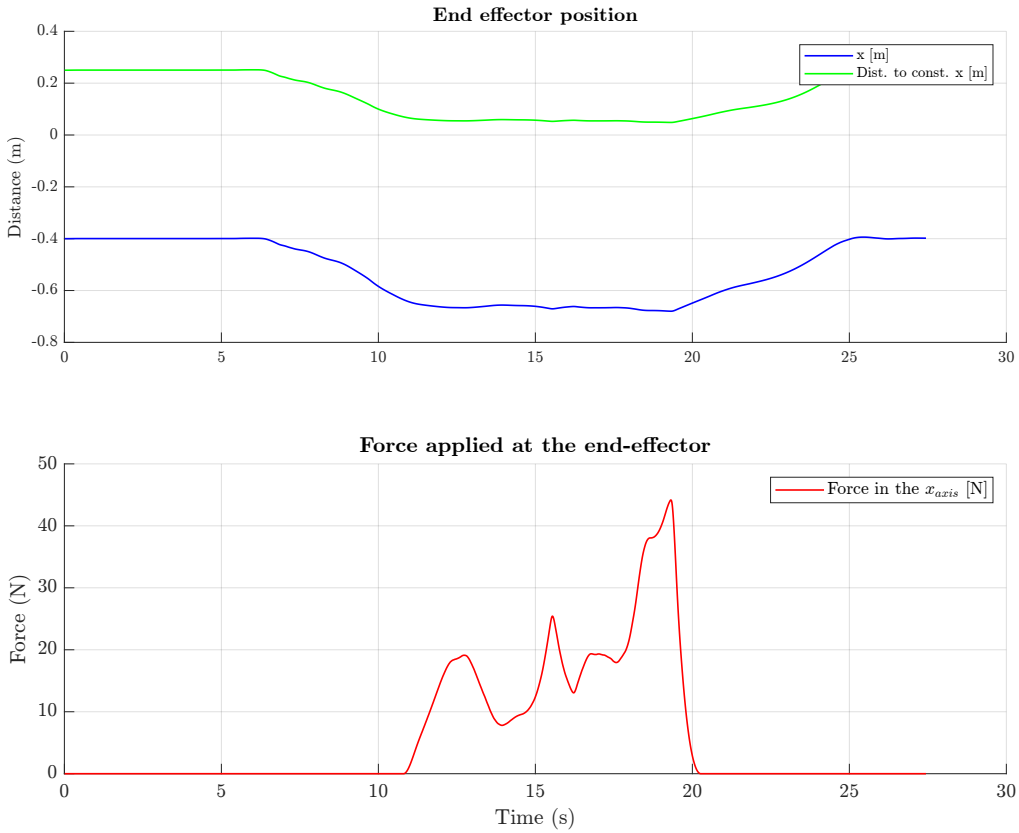


Fig. 4.5: End-effector's trajectory along the X axis in the first figure, followed by the Cartesian force applied onto the end-effector on the same trajectory.

The same way it occurs with the previous virtual constraint, the further the KUKA is pushed towards the constraint the higher the force applied is, but never surpassing the mark of 0.7 m in the negative x axis. In the current test, the positioning of the KUKA LBR iiwa is similar to the one shown in Figure 4.3 but on the opposite direction.

Then, the y axis is to be evaluated on matching planes as in the x axis but curved 90º, due to the base of the model being a square. Hence, it is also to be expected two planes at $y = 0.7$ and $y = -0.7$.

Plane constraint at y = 0.7

Similarly to the previous axis constraints, the virtual planes at the y axis demonstrates how the force increases in the opposite direction the end-effector is colliding to when encountering the constraint. For the next example in Figure 4.6, several pushes towards the constraints produces some oscillations in the second graph as the constraint repels the end end-effector.

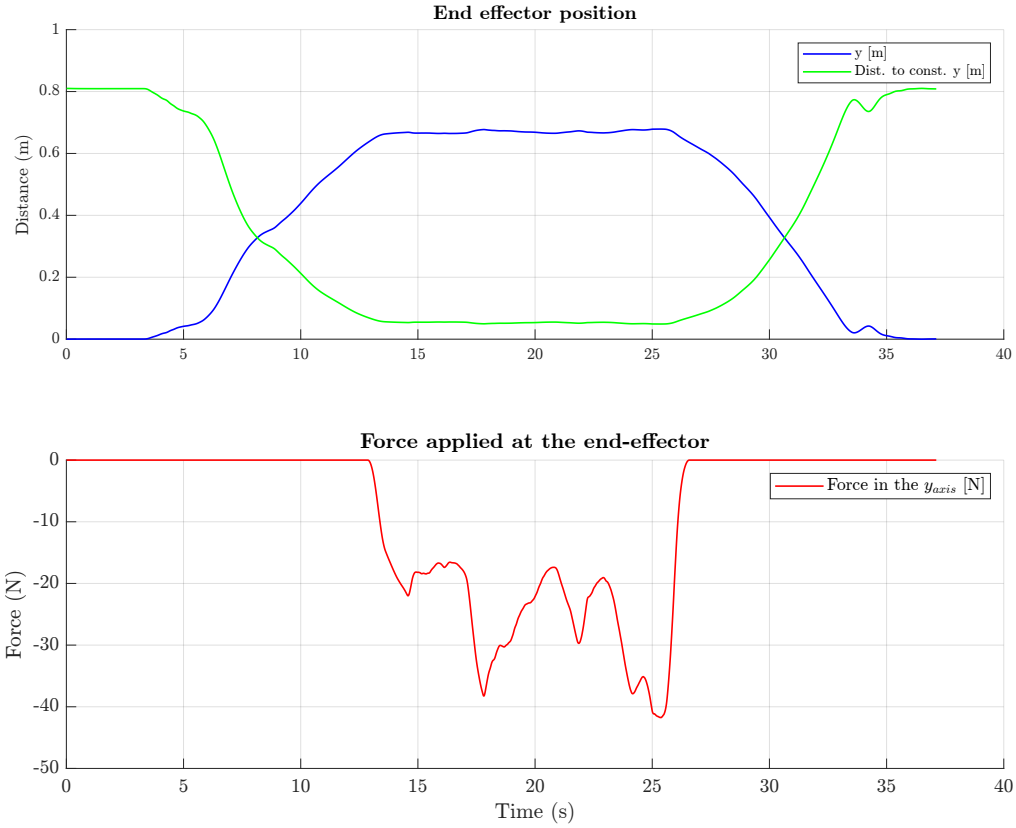


Fig. 4.6: End-effector's trajectory along the Y axis in the first figure, followed by the Cartesian force applied onto the end-effector on the same trajectory.

The same way it is illustrated for the planes in the negative y axis in Figure 4.7.

Plane constraint at y = -0.7

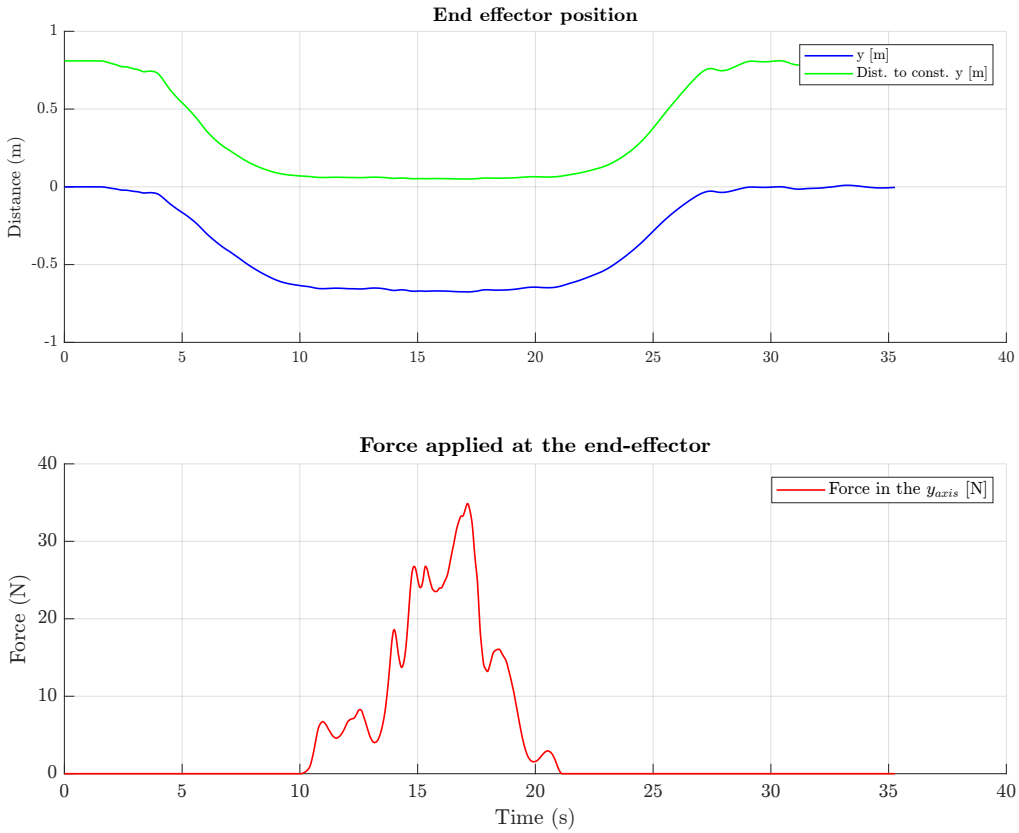


Fig. 4.7: End-effector's trajectory along the Y axis in the first figure, followed by the Cartesian force applied onto the end-effector on the same trajectory.

It can also be observed that for all presented figures there exists a direct or indirect relation in between the position of the end-effector and the distance to the constraint. This is worth showing as the distance to the constraint is proved to not being violated in any case by not exceeding a negative distance to the constraint of target.

To finalise the tests, the constraints in the z axis are presented. As imposed by the 3D model, a wall on top of the robot is reached at $z = 1$ shown in Figure 4.8.

Plane constraint at z = 1

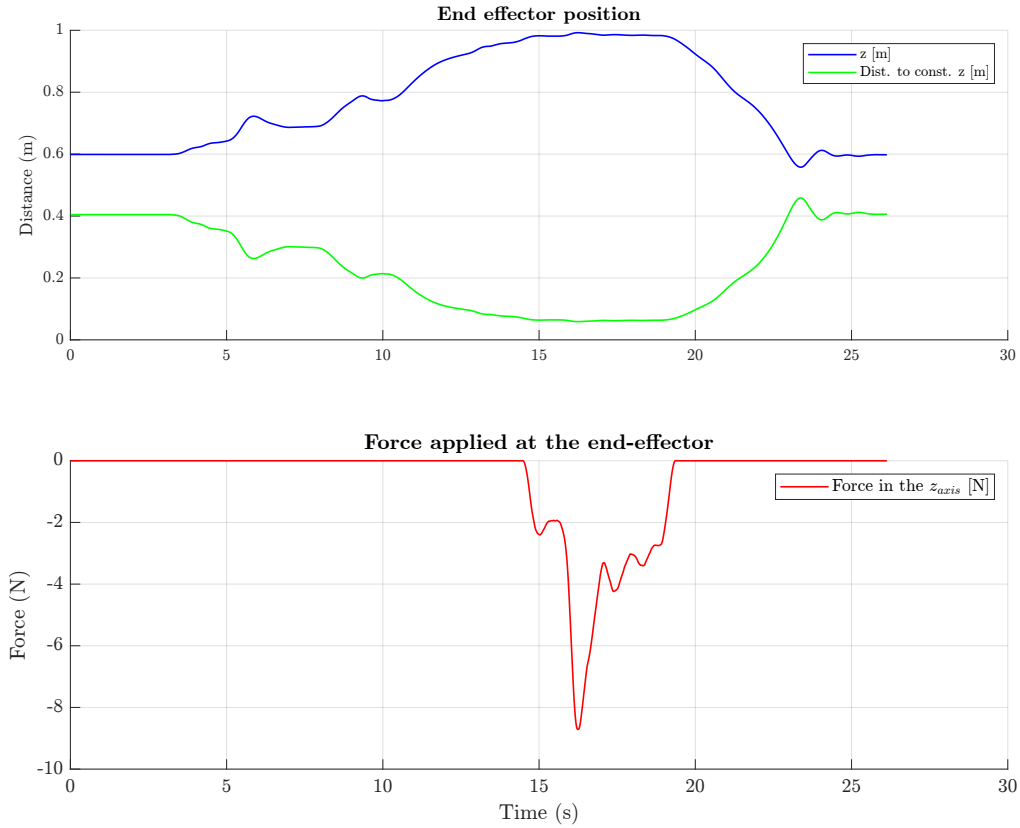


Fig. 4.8: End-effector's trajectory along the Z axis in the first figure, followed by the Cartesian force applied onto the end-effector on the same trajectory.

In the Figure 4.8 it is illustrated the collision of the manipulators end-effector to the ceiling constraint placed at z = 1m. The behaviour is again equal as for the previous collisions.

Plane constraint at z = 0

Lastly, not because the model imposed it but because it was indeed hard-coded, a constraint was placed at $z = 0$ with some security margin of 0.2m. This was done as there is a known surface in the actual robot that is not accounted for in the 3D model created, and it is mainly for the robot not to hit the real table. The details are further

explained in 5.1, but the main reason behind this is simplicity due to a lack of time for the real tests scenario. It has already been mentioned earlier in the section that the bottom model constraint was discarded as the robot was unable to hit it, and thus, this constraint will not be shown in the current chapter.

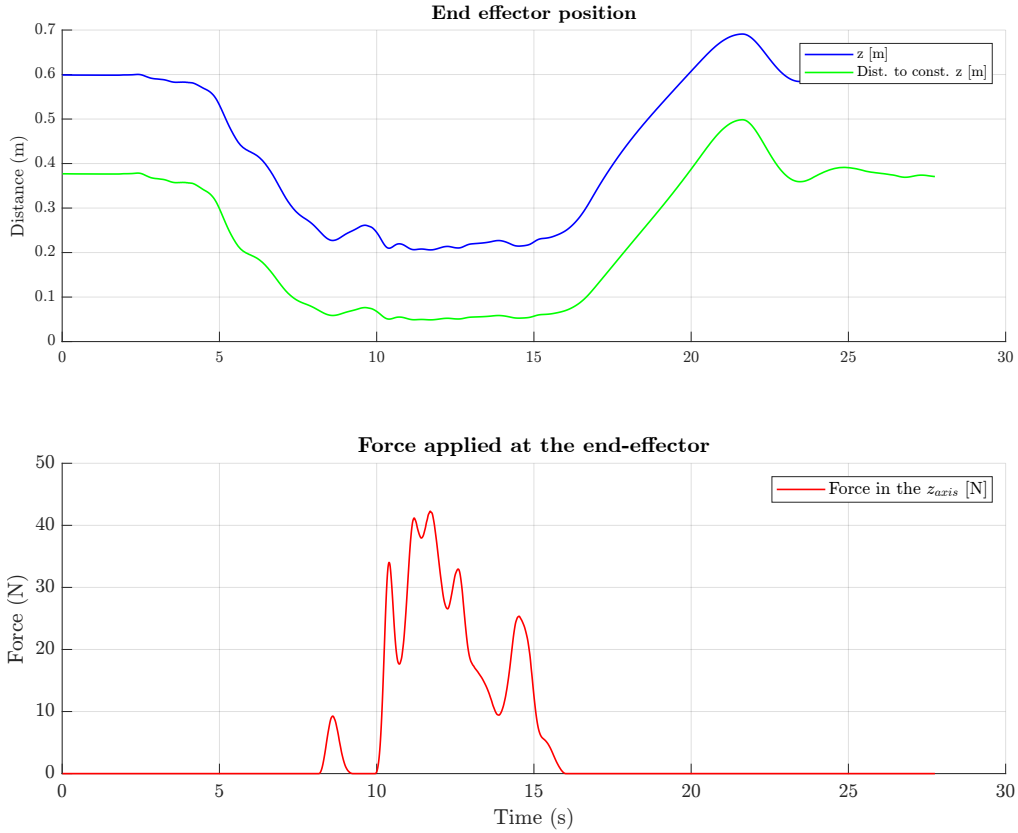


Fig. 4.9: End-effector's trajectory along the Z axis in the first figure, followed by the Cartesian force applied onto the end-effector on the same trajectory.

Multi constraint collision

The Figure 4.11 presents the behaviour of the system when colliding with more than one constraint at a time and proves the third requirement stated at the beginning on the Chapter 4.

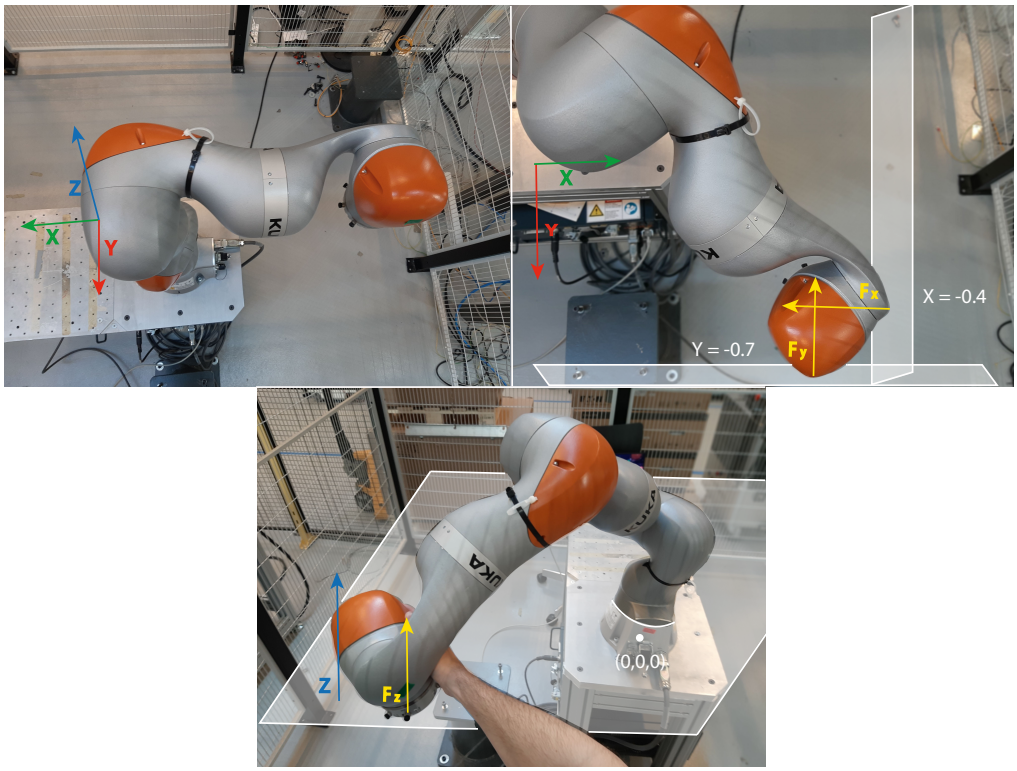


Fig. 4.10: Illustration of the test where the manipulator collides with 3 constraints at a time.

In this figure, all 3 axis are relevant in terms of showing the increment of Cartesian force applied in all directions as the robot collides with 3 planes at a time, each situated on one different axis: $x = -0.4$, $y = 0.7$, $z = 0$. It is worth noticing that due to the model placement, the collision with a corner of the box was never hit; therefore, the plane situated in the $x = -0.7$ was displaced 20 cm in the positive x_{axis} becoming a constraint in the $x = -0.4$.

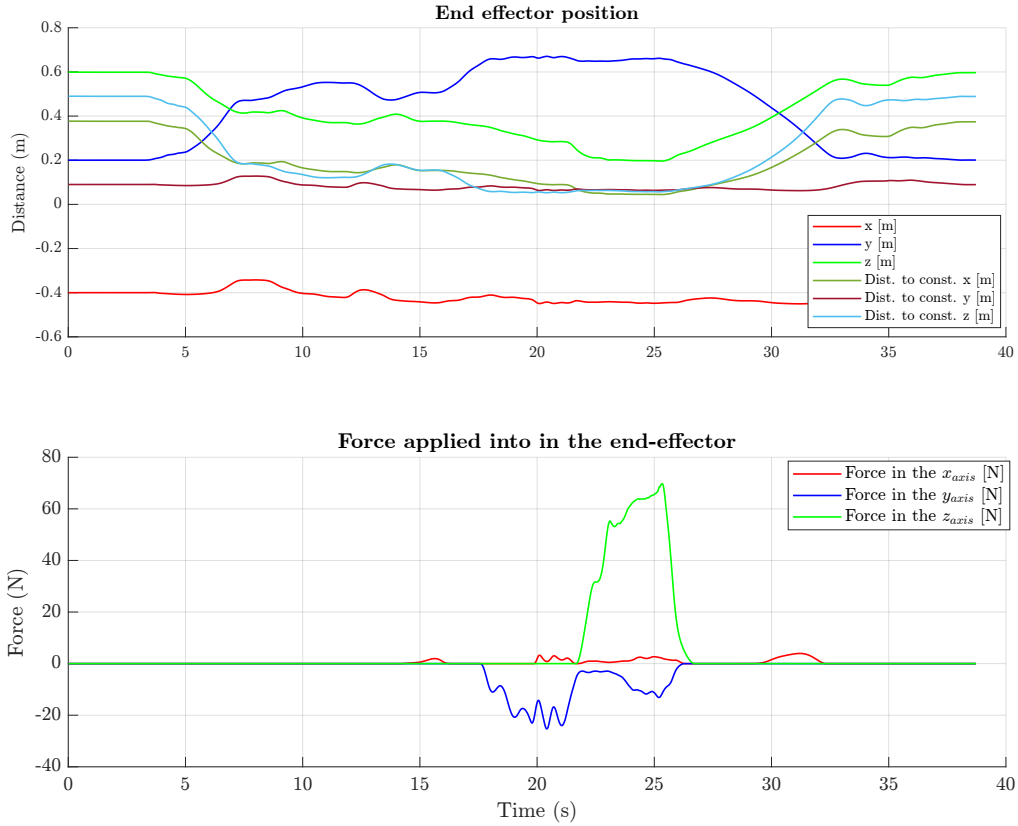


Fig. 4.11: Multi plane collision

4.2.1 Results evaluation

If we recap on the objective once again, it is aimed to automatize the implementation of virtual limitations that command the controller to behave as desired by the model. To validate the implementation of the system studied in the current report, a set of requirements is listed and several tests were performed that are expected to fulfill those requirements.

In the end, looking at all figures in the tests chapter, it can be observed how the distance to the constraint to test is never violated by the serial manipulator, staying within the limits imposed. Moreover, at the same time the constraint is approached, the forced is exerted on the end-effector of the robot arm increases due to the virtual collision the arm is suffering on the controller side. This can also be deducted from the different

pushes the robotic arm does against the wall in an attempt to violate the constraints. Finally, a test on the collision with more than 1 limitation is examined in 4.2, proving the effectiveness of the system while handling not only 1 but every limitation imposed at a time.

It can then be stated that the system performs within the requirements, and it has been proved the difficulty of the system violating the constraints while inputting an entire 3D model of the environment to the system.

Chapter 5

Conclusion and Future Work

The last chapter of the report focuses on discussing the various inconvenients encountered while developing the application and performing the different tests, comparing the actual results with the expected ones initially. Furthermore, it revises in the section 5.3 the actual outcome, determining whether the requirements set in the tests Chapter 4 were fulfilled. Lastly, the possible future work is presented.

5.1 Discussion

In a further analysis of the results, the more obvious difference is in the output of the Z axis constraints. The top constraint at $z = 1$ is almost hit despite the distance calculated to the constraint is not ever reaching 0 and thus, not hitting. This might be caused by an imperfect model of the box, either by not being the exact measures it has been previously mentioned or there is a margin error while transforming the mesh to point clouds and this causes the plane detection algorithm to impose the constraint a bit further up. All in all, for a real world scenario the common application will be to implement a security margin higher than the constraint activates on and repels the projected point, for instance, 10 centimeters or more would ensure the manipulator will never hit the actual environment.

In the same mentioned axis, a hard-coded constraint in the $z = 0$ plane was needed for the KUKA LBR iiwa not to hit the table it is located on. Such specific model was not created from the start as the software it was generated in (CloudCompare) lacks of enough tools to do so. Indeed, the positioning of the origin couldn't successfully be

modified either in this software or others such as SolidWorks.

Additionally, the time factor has been crucial for this project. The knowledge behind the control software taking care of repelling the robots end-effector based on the wrench modelling and the twist theory revised in [1] is exigent enough for the start of the actual development of the current solution to be delayed. An evaluation of all the concepts the controller is handling in [1] may be also of interest to prove the system performs as expected. This has been assumed for the current report as the controller has not been changed but rather some computations have been added to it additional to the previous work.

A good supplementary work for the tests presented would also have been able to simulate the 3D environment at the same time the robot moves in the real scenario. This would show visually what is illustrated with the use of plots in the Cartesian space to imagine the real behaviour of the manipulator.

5.2 Future Work

Some possible additions and supplementary work have already been covered previously in the report, but this section will enumerate on the possible future works that the project may follow.

Due to the lack of time of the practical work, the first thing worth proving is obtaining similar outputs on the controller side as the ones obtained in [1], validating the final implementation in terms of the controller side, despite that it is already proved to repel constraints as it has been done for the current project. Similarly, it would be practical to simulate the box at the same time as the robot moving in *Rviz* or a similar program verifying the correct placement of it in space through the software proposed.

In the same path, the curved constraints implementation are not tested out but a logic was created for its handling with the suggested system; its logic presented in 3.4 lacks of actual code implementation. Additionally, this method is assuming every surface can be defined as bigger or smaller combination of planes. Further study on this topic may be done to come up with a more efficient architecture. Besides, there are entire studies about the projection of a point onto surfaces that may be revised further [41], [42].

In terms of the current system presented, different scales may be compared on the plane extraction algorithm depending on the case of use; as well as different model inputs being used.

In case you have questions, comments, suggestions or have found a bug, please do not hesitate to contact me. You can find my contact details below.

5.3 Conclusion

In this project, an adaptive solution for the implementation of virtual trajectory limitations for a 7-degree-of-freedom manipulator has been presented. The solution includes the study of the previous controller designed and the concepts involved, the study of the different approaches in literature for plane extraction and the combination of the two techniques for the creation of a system capable of not only handling planar surfaces but any kind of surface and constraint the robot may encounter in a 3D model input.

For doing this, a KUKA LBR iiwa industrial manipulator was utilised in the laboratory of the University. A controller studied in [33] has been implemented along with a multi-plane detection algorithm [4] that is capable of analyse planes at different scales. Different scripts in both C++ and python were created for the handling of the point cloud model into the plane segmentation software and those constraints imposition into FRI application of the KUKA controller.

The system has finally been proved to work for each constraint of the tested model used as input in Chapter 4, in this case, a box-like point cloud from a mesh was used, as described in Chapter 2. The solution presents itself a robust application for the utilisation of the workplace digital modelling as constraints that can be directly inserted in the controller.

Further research may include testing of curved surfaces models such as the one presented in 3.4 among others, detailed in 5.2.

Antonio Aranzana Sánchez
email: aaranz20@student.aau.dk
Friederik Bajers Vej 7
9220 Aalborg Ø

Bibliography

[1] S. Hjorth. *Investigation and Implementation of Workspace restrictions for the KUKA LBR iiwa*. June 2019.

[2] R. M. Murray, Z. Li, and S. S. Sastry. "A Mathematical Introduction to Robotic Manipulation". In: *CRC Press* (1994).

[3] G. Raiola et al. "Development of a Safety- and Energy-Aware Impedance Controller for Collaborative Robots". In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 1237–1244. DOI: 10.1109/LRA.2018.2795639.

[4] T. Lejemble et al. "Persistence Analysis of Multi-scale Planar Structure Graph in Point Clouds". In: *Computer Graphics Forum* 39.2 (2020), pp. 35–50. DOI: https://doi.org/10.1111/cgf.13910.

[5] M. Stevens. "Pros and cons of using industrial robots in your manufacturing operation". In: (Sept. 2021). URL: https://tinyurl.com/22prpfka.

[6] M. Knudsen and J. Kaivo-oja. "Collaborative Robots: Frontiers of Current Literature". In: *Journal of Intelligent Systems: Theory and Applications* 3 (June 2020), pp. 13–20. DOI: 10.38016/jista.682479.

[7] J. Haviland and P. Corke. "Maximising Manipulability During Resolved-Rate Motion Control". In: (Feb. 2020).

[8] F. J. Abu-Dakka and M. Saveriano. "Variable Impedance Control and Learning—A Review". In: *Frontiers in Robotics and AI* (Dec. 2020). DOI: 10.3389/frobt.2020.590681.

[9] N. Hogan. "Impedance Control: An Approach to Manipulation". In: *1984 American Control Conference*. 1984, pp. 304–313. DOI: 10.23919/ACC.1984.4788393.

[10] C. Scoccia et al. "Real-Time Strategy for Obstacle Avoidance in Redundant Manipulators". In: (Jan. 2021), pp. 278–285. DOI: 10.1007/978-3-030-55807-9_32.

[11] M. Li et al. "Adaptive Control of Robotic Manipulators With Unified Motion Constraints". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47.1 (2017), pp. 184–194. DOI: 10.1109/TSMC.2016.2608969.

[12] N. Paperno et al. “A Predictive Model for Use of an Assistive Robotic Manipulator: Human Factors Versus Performance in Pick-and-Place/Retrieval Tasks”. In: *IEEE Transactions on Human-Machine Systems* PP (Aug. 2016). DOI: 10.1109/THMS.2016.2604366.

[13] S. Trimble et al. “Context-aware robotic arm using fast embedded model predictive control”. English. In: *Irish Systems and Signals Conference: Proceedings*. Context-aware robotic arm using fast embedded model predictive control. IEEE, June 2020. DOI: 10.1109/ISSC49989.2020.9180217.

[14] J.-H. Chen and K.-T. Song. “Collision-Free Motion Planning for Human-Robot Collaborative Safety Under Cartesian Constraint”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 4348–4354. DOI: 10.1109/ICRA.2018.8460185.

[15] C. T. Landi et al. “Safety Barrier Functions for Human-Robot Interaction with Industrial Manipulators”. In: *2019 18th European Control Conference (ECC)*. 2019, pp. 2565–2570. DOI: 10.23919/ECC.2019.8796235.

[16] F. Chen et al. “Evolutionary artificial potential field method based manipulator path planning for safe robotic assembly”. In: *2009 International Symposium on Micro-NanoMechatronics and Human Science*. 2009, pp. 92–97. DOI: 10.1109/MHS.2009.5352075.

[17] P. Chotiprayanakul et al. “A 3-Dimensional Force Field Method for Robot Collision Avoidance in Complex Environment”. In: Sept. 2007. DOI: 10.22260/ISARC2007/0026.

[18] K. P. Tee, R. Yan, and H. Li. “Adaptive admittance control of a robot manipulator under task space constraint”. In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 5181–5186. DOI: 10.1109/ROBOT.2010.5509874.

[19] M. Rauscher, M. Kimmel, and S. Hirche. “Constrained robot control using control barrier functions”. In: Oct. 2016, pp. 279–285. DOI: 10.1109/IROS.2016.7759067.

[20] J. D. M. Osorio et al. “Physical Human-Robot Interaction under Joint and Cartesian Constraints”. In: *2019 19th International Conference on Advanced Robotics (ICAR)*. 2019, pp. 185–191. DOI: 10.1109/ICAR46387.2019.8981579.

[21] L. M. Surhone, M. T. Tennoe, and S. F. Henssonow. *Transformation Matrix*. Betascript Publishing, Aug. 2010, p. 88. ISBN: 613123079X.

[22] M. B. Popovic and M. P. Bowers. *Kinematics and Dynamics*. Ed. by M. B. Popovic. Academic Press, 2019, pp. 11–43. ISBN: 978-0-12-812939-5. DOI: 10.1016/B978-0-12-812939-5.00002-1. URL: https://www.sciencedirect.com/science/article/pii/B9780128129395000021.

[23] C. A. D. Cardenas. “Development of a safety-aware intrinsically passive controller for a multi-DOF manipulator”. In: 2017.

[24] B. Siciliano and O. Khatib, eds. *Springer Handbook of Robotics*. Spinger, 2008, p. 1611. ISBN: 978-3-540-30301-5. DOI: 10.1007/978-3-540-30301-5.

[25] B. E. Rapp. "Chapter 7 - Vector Calculus". In: *Microfluidics: Modelling, Mechanics and Mathematics*. Ed. by B. E. Rapp. Micro and Nano Technologies. Oxford: Elsevier, 2017, pp. 137–188. ISBN: 978-1-4557-3141-1. DOI: 10.1016/B978-1-4557-3141-1.50007-1.

[26] L. M. Kevin and F. C. Park. "MODERN ROBOTICS MECHANICS, PLANNING, AND CONTROL". In: Cambridge University Press, May 2017, p. 642. ISBN: 9781107156302.

[27] S. R. Ploen and F. C. Park. "A Lie group formulation of the dynamics of cooperating robot systems". In: *Robotics and Autonomous Systems* 21.3 (1997). Critical Issues in Robotics, pp. 279–287. ISSN: 0921-8890. DOI: https://doi.org/10.1016/S0921-8890(96)00802-0.

[28] S. Stramigioli. *Modeling and IPC Control of Interactive Mechanical Systems - A Coordinate-Free Approach*. Springer, 2001, p. 280. ISBN: 978-1-85233-395-9. DOI: 10.1007/BFb0110400.

[29] C. Ott. *Cartesian Impedance Control of Redundant and Flexible-Joint Robots*. Springer Berlin, 2001, p. 192. ISBN: 978-3-642-08873-5. DOI: 10.1007/978-3-540-69255-3.

[30] S. Stramigioli. "From differentiable manifold to interactive robot control". In: PhD thesis, Dec. 1998. ISBN: 90-9011974-4.

[31] C. Secchi, C. Fantuzzi, and S. Stramigioli. *Control of Interactive Robotic Interfaces*. Vol. 1. Springer-Verlag Berlin Heidelberg, 2007, p. 233. ISBN: 978-3-540-49715-8. DOI: 10.1007/978-3-540-49715-8.

[32] O. Khatib. "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots". In: 5 (Mar. 1986), pp. 90–98. DOI: 10.1177/027836498600500106.

[33] S. Hjorth et al. "An Energy-based Approach for the Integration of Collaborative Redundant Robots in Restricted Work Environments". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2020, pp. 7152–7158. ISBN: 978-1-7281-6213-3. DOI: 10.1109/IROS45743.2020.9341561.

[34] I. Realsense. *Intel Realsense D400 Series Product Family*. Jan. 2019. URL: shorturl.at/ginFI.

[35] C. Liu et al. "PlaneRCNN: 3D Plane Detection and Reconstruction from a Single Image". In: (Jan. 2019).

[36] Z. Yu et al. "Single-Image Piece-wise Planar 3D Reconstruction via Associative Embedding". In: (Apr. 2019).

[37] D. S. Liu et al. *3D Point Cloud Analysis*. Springer International Publishing, 2021. ISBN: 978-3-030-89179-4. DOI: 10.1007/978-3-030-89180-0_1.

[38] J.-E. Deschaud and F. Goulette. "A Fast and Accurate Plane Detection Algorithm for Large Noisy Point Clouds Using Filtered Normals and Voxel Growing". In: *3DPVT*. Paris, France, May 2010. URL: https://hal-mines-paristech.archives-ouvertes.fr/hal-01097361.

[39] L. Lamport. “How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs”. In: *IEEE Transactions on Computers C-28* 9 (1979), pp. 690–691.

[40] A. Boulch and R. Marlet. “Fast and Robust Normal Estimation for Point Clouds with Sharp Features”. In: *Comput. Graph. Forum* 31.5 (2012), 1765–1774. ISSN: 0167-7055. DOI: 10.1111/j.1467-8659.2012.03181.x.

[41] S.-M. Hu and J. Wallner. “A second order algorithm for orthogonal projection onto curves and surfaces”. In: *Computer Aided Geometric Design* 22.3 (2005), pp. 251–260. ISSN: 0167-8396. DOI: https://doi.org/10.1016/j.cagd.2004.12.001.

[42] H.-C. Song et al. “Projecting points onto planar parametric curves by local biarc approximation”. In: *Computers & Graphics* 38 (2014), pp. 183–190. ISSN: 0097-8493. DOI: https://doi.org/10.1016/j.cag.2013.10.033.

Appendix A

Implementation

The Appendix A focuses on describing the general setup and communication between the different devices and tools used in the current project: the KUKA LBR iiwa manipulator connected to *KUKA's Robot Controller* (KRC) and the computer generating the control through the Fast Research Interface (FRI).

A.1 KUKA LBR iiwa R800

It is mentioned throughout the project the manipulator of use is the KUKA LBR iiwa R800. This collaborative robot is used as it is one of the free available robots that the Manufacturing and Production department from the AAU could provide at the moment, which counts on position/torque sensors and torque controlled motors for each joint. Additionally, it was also the robot used in the previous studies the current project is based on [1], [33].

A.1.1 KUKA Robot Controller - KRC

The manipulator of study is controlled with the use of the KRC, which is responsible for the transmission control inputs at the same time as it reads the data of the integrated sensors. To be able to build a control application using the FRI library from KUKA, one needs to first upload a Java application into the KRC via the KUKA Line Interface (KLI) port and Sunrise Workbench in a windows machine. This process will do the KRC running natively, thus any other application connected to the Java application will do as well.

For creating such application interaction, an FRI User Datagram Protocol (UDP) client is created on a Linux-based machine. In this specific case, a Windows computer with the Windows Subsystem for Linux 2 (WSL2) tool was used. The Linux environment connects then to the KRC via the UDP connection.

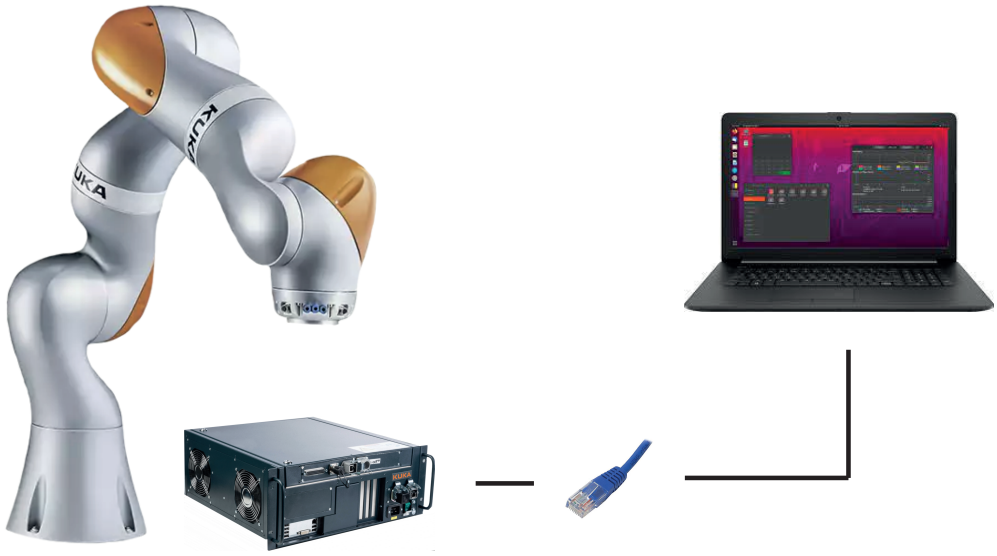


Fig. A.1: Caption

A.1.2 Fast Research Interface - FRI

The KUKA FRI library provides real-time interfacing between an FRI server application running inside the KRC and the client created on the external Linux system via UDP. This application needs to be specifically configured such that it overrides the control signal sent by the FRI C++ client, overlaying its functionality and adding the new logic. The new control signals sent through the application could either be positions, torques or wrenches depending on the specified Java application onto the KRC.

A.2 Code description

The first thing that will run on the code is the analysis of the points inside the planes found. An exhaustive pre-processing of the 3D model to implement is assumed to be done prior to the execution of the program as described in Chapter 3. Once some planes are found by the software, the are needed to be used as input for the next software step.

The analysis and implementation of constraints is done the very first thing when running the code, based on the results of the pre-processing stages. After, the program will focus on controlling the input τ such that the end-effector handles to stay within the Cartesian constraints and the join limit avoidance strategy explained in Chapter 2. The procedure is as follows:

1. The repelling Wrench $W_{C_j}^{0,i}$ is computed for the i^{th} link, in this case being the end-effector, for every constraint imposed by the system. If the euclidean distance of the end-effector to a constraint C is not sufficient for the constraint to activate, the Wrench is expected to be 0.

2. The reactive control is then executed, where the motion generating torque τ_{ctrl} based on the displacement H_d^{tcp} and the repelling Wrenches from active constraints are computed.

3. The torques resulting from the Joint-limit-avoidance scheme is also calculated, along with the total torque τ^T resulting from eq. 2.24.