Adaptive trajectory generation with workspace restictions for clinical applications

Carolina Gomez Salvatierra

Master Thesis - Robotics

3rd June 2021

Attributions

This document was typeset using LaTeX.

AALBORG UNIVERSITY
DENMARK

Master Thesis - Robotics

Fredrik Bajers Vej 7
9000 Aalborg

Title:

Adaptive trajectory generation with workspace restrictions for clinical applications

Project Period:

3rd February - 3rd June, 2021

Project Group:

Group 1063b

Author:

Carolina Gomez Salvatierra

Supervisors:

Dimitris Chrysostomou

Sebastian Hjorth

Number of Pages: 64
Date of completion: 03-06-2021

Synopsis:

Collaborative robotics and healthcare have progressed in the last decades, being now in a wide topic with multiple branches, such as surgery, laboratory robots, exoskeletons, prosthesis, therapy and much more.
In the case of therapy, collaborative robots are already in use for rehabilitation and available for different application types, e.g. for improving the rehabilitation in lower or upper limbs.

In this project the focus relies on the workspace restrictions on rehabilitation exercises, in order to prevent the patient from doing exercises trajectories incorrectly and potentially deviating the manipulator from its goal position towards an non-allowed space. Overall, the workspace restrictions have been done using an approach based on virtual constraints and in case of not letting the patient deviate the robot, it is based on the cooncept of Artificial Potential Field.

From a practical point of view, the project was developed in order to improve the robotic platform called ROBERT©, a collaborative assistant used by therapists, mostly on the recovery from stroke patients.
ROBERT© is a platform with a Kuka LBR Med on the top, which has an end-effector that can be attached to the patient's leg with a tool. Afterwards, the robot helps the patient to complete physical routines.
The main goal of this project is to provide a proof of concept under a new perspective on path planning for manipulators in clinical applications.

Summary

Hereby the presented project is within healthcare and collaborative robotics. Collaborative robotics are a wide field within research that covers many areas and in the specific case of healthcare, there are different branches, such as therapy robotics, medical or surgical assistants.

In the case of this project, the focus is on assistive therapy, where collaborative robots aim to improve a patient's rehabilitation process. More concretely, this projects presents a new concept for trajectory generation and workspace restriction implementation mainly based on Artificial Potential Field(s) (APF).

From a practical point of view, this project is also meant to solve a real world problem, having a specific use case for the ROBERT platform, property of Life Science Robotics (LSR) Aps.

Overall, the proposed approach has three main elements: a trajectory generator, a set of virtual constraints and a non-dimensional APF that restricts the robot motion. In every chapter, the three elements are analysed under a different perspective. For example: Chapter 4 the three elements are described on a mathematical manner, or in Chapter 6, the three elements are tested and the results are evaluated.

To summarize, this documentation describes the research done within the field, the presented approach and overview as a whole and separately in detail.

Afterwards, the report goes into detail about the implementation and testing, including the algorithm design, testing process and results.

Finally, the report reviews the project as a research concept and the obtained results, which leads to a conclusion and potential future projects.

Preface

This report was written by group 1063 of 10th Semester Robotics, at Aalborg University.

In order to have a better understanding about the project, see the following reading guidelines:

- The context and problem analysis are explained in this introduction and in the next chapter, which are Chapters 1 and 2.
- Chapters 3 and 4 explain the chosen approach for the problem and the methodology for the solution.
- The application of the chosen methods on the robot is described on Chapter 5.
- The testing process and outcomes are described on Chapter 6.
- The last three Chapters 7, 8 and 9 sum up the project, highlights the most important results and adds the possible future works for further development.

All the simulation, modelling and relevant software elements are available in this project's repository [5].

Carolina Gomez Salvatierra

Contents

Abbreviations List

APF	Artificial Potential Field(s).
API	Application Programming Interface.
CAD	Computer-Aided Design.
CS	Cartesian space.
DOF	Degree(s) of Freedom.
FK	Forward Kinematics.
HRC	Human-Robot Collaboration.
IDE	Integrated Development Environment.
IK	Inverse Kinematics.
JS	Joint space.
LSR	Life Science Robotics.
RDO	Research and Development Objective.
TCP	Central Tool Point.

Nomenclature

Robot modelling

n_θ	Kuka LBR Med amount of degress of freedom	7
$\theta, \dot{\theta}, \ddot{\theta}$	Joints angle, velocity and acceleration $n_\theta \times 1$ vectors	
τ	Joints torque $n_\theta \times 1$ vector	
F	Force matrix at the end-effector in Cartesian space	
M	Inertial force $n_\theta \times n_\theta$ matrix	
C	Coriolis and centrifugal forces $n_\theta \times n_\theta$ matrix	
G	Gravitational force $n_\theta \times 1$ matrix	
J_a	Analytical Jacobian $6 \times n_\theta$ matrix	
T_i^j	Transformation 4×4 matrix from frame i to frame j	
x, x_i	Position in Cartesian space, format 3×1 vector	
x_{ee}	Position of the end-effector in Cartesian space, format 3×1 vector	
$\dot{x}$, $\dot{x}_i$	Velocity in Cartesian space, format 3×1 vector	
f_{ee}	Force at the end-effector in Cartesian space, format 3×1 vector	

Trajectory generation terms

$\dot{x}_r$	Reference or desired velocity in Cartesian space, format 3×1 vector	
e	Euler's constant	2.718281

Virtual constraint terms

x_{vc}, $\vec{x_{vc}}$	Point from a virtual constraint in Cartesian space, format 3×1 vector
x_n, $\vec{x_n}$	Updated path point in Cartesian space, format 3×1 vector
r_{vc}	Constant radius of a virtual tube

α Angle in a virtual tube section from a path point to virtual constraint, being both contained in the same normal plane

Artificial Potential Field terms

λ Potential ratio

β Velocity ratio

δ,δ_β Proportional margin distance towards a virtual constraint

ρ Absolute distance towards a virtual constraint

x_{center}, x_{goal} Reference position in Cartesian space, format 3×1 vector

k_{cs}, d_{cs} Stiffness and damping gains in Cartesian-Space

List of Figures

List of Tables

1 Introduction

In the last years, technology has been developed exponentially. This is due to how previous key technological advances (mass production, computers [1]) have given easy access to tools and knowledge. Parallel to that, society has also changed and this meant new needs in demand on the market.

One of the most remarkable new demands is the safety and ergonomics at the workplace [24], in order to prevent potential accidents and improve the workspace conditions for operators. As a consequence, research has gone further in Human-Robot Collaboration (HRC) and the new concept of cyber-physical environment was introduced, which is the use of devices and new smart technologies that can interact with humans and adapt to their needs and other unexpected events around them.

Taking into account how society and the market have reached this situation, it is possible to understand the lastest changes in robotics, where a new generation of robots has been introduced: collaborative robots, or cobots. Compared to the previous generations, like the conventional industrial manipulators, where robots and workers were separated by fences and other safety measures, cobots are safe for workers and can share the same workspace. Collaborative robots work together with humans in order to complete tasks and achieve common goals.

Outside the industry, technology development and robotics have also an important role in other fields, for example in food, social services, security or healthcare. In case of collaborative robotics, cobots are nowadays a wide research topic with a high variety of use cases and applications.

Going into more detail on the healthcare collaborative robotics, at the moment there are different types of cobots, such as exoskeletons, prosthesis controllers or surgery assistants.

The focus of this project is on collaborative robotics in healthcare, specifically into rehabilitation robotics. In this concrete area, collaborative robots aim to help both the physiotherapist and the patient through the rehabilitation process.

Use case

This project is oriented to solve a current real-world problem with an specific robotic platform. This platform is called ROBERT© and it belongs to the company Life Science Robotics (LSR) ApS, located in Aalborg (Denmark).

ROBERT is a collaborative platform used in clinics by physiotherapists with patients that are recovering their mobility in their lower limbs. The platform works as an assistant that takes

care of the leg weight of the patient during rehabilitation exercises, reducing the amount of work load of the physiotherapist. On the other hand, the physiotherapist shows by demonstration the exercise trajectory to the robot, sets the exercise parameters and supervises the exercises on the run.

As shown in Figure 1.1, ROBERT has different components and devices. The most relevant ones are:

- Robot arm: a redundant manipulator, model Kuka LBR Med. It has seven Degree(s) of Freedom (DOF).
- Linkage: a specific end-effector that allows the arm to be attached and de-attached to the leg of the patient. It also has two buttons for the hand guiding and attaching functionalities.
- Control cabinet: the robot controller. It is located inside the platform and makes the robot mobile, so the user can push the platform and use it in different locations.

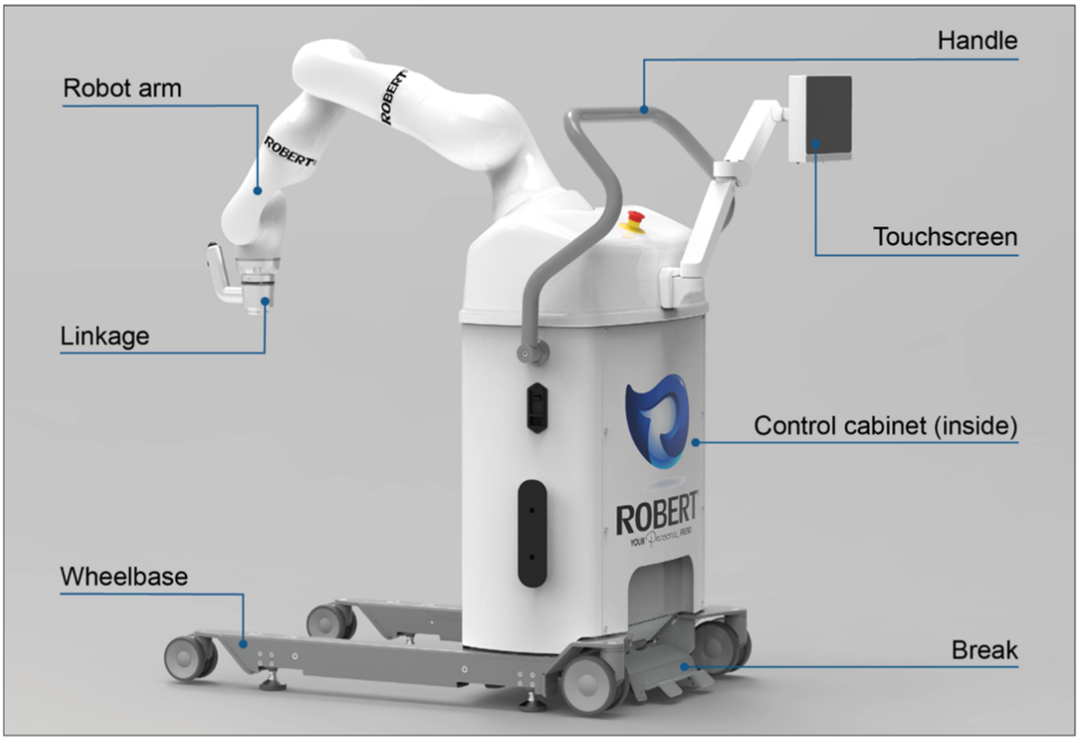

Fig. 1.1: ROBERT platform overview. Image taken from [15].

Going now into the use application, the physiotherapist must teach the trajectory to the robot by demonstration at the beginning of every exercise. Afterwards, the physiotherapist can program different exercise types on the robot. The main idea is that an exercise is divided in sets with a break time in between. Then, every set has an amount of repetitions that the patient and

robot must complete. A repetition is the execution of the demonstrated trajectory from the start to the end point.

There are three different types of exercises:

1. Passive exercises: the patient cannot move on his own and the robot takes fully care of moving the limb. This type of exercise is meant for an early recovery stage.

   The manipulator activates a position controller and executes the recorded trajectory exactly as demonstrated. The recording process of the demonstration saves the current position, orientation, velocity and force every $100ms$. The force feedback is only used to detect if the patient aims to stop the trajectory execution or if the physiotherapist is intervening.

2. Active exercises: the patient can move his limbs and has to move the leg from beginning to end and push the robot along with it. This type of exercise is meant for mobility improvement.

   The manipulator in this case activates an impedance controller, which works as a spring and has a resistance against the force of the patient. The resistance level of the robot is set by the physiotherapist (from 1 to 10). The recording process of the demonstration saves only the start point and the end point, and it interpolates a straight line between them. When the patient stops pushing the robot, the robot shall go to the closest point in the straight line.

3. Hybrid exercises: it combines the previous two types. The physiotherapist chooses which sets and repetitions are done using the passive mode and which using he active mode.

Currently this setup has some disadvantages and different aspects that can be improved:

- The trajectory recording cannot be rectified. If the physiotherapist makes a mistake during the recording, the recording must be repeated.
- Since the trajectory must be demonstrated every time, it makes sense that the demonstrated trajectory does not contain obstacles on the path, and therefore, it is valid that the robot has to repeat it right afterwards on the exercise execution.
- As a consequence from the previous point, there is a lack of methods for obstacle avoidance or path planning to adjust the trajectory to changes on the environment and the robot workspace only concerns the manipulator constraints.
- The trajectory is fully demonstrated by the physiotherapist, which means that the performance quality depends entirely on the physiotherapist's knowledge and skills. In case of an inexperienced physiotherapist, this could reduce the benefits of using the ROBERT platform on a patient.

This list of points is the main reference to formulate the problem and requirements to solve in this project. The solution shall be focused on approaching these issues and developing one or more methods to improve the current system.

2 Problem analysis and formulation

2.1 Previous works

Healthcare robotics

The presented problem concerns robotics within healthcare. This field of research includes different areas, such as rehabilitation robots, phrostetics and exoskeletons, laboratory robots and rehabilitation robots.

A review on the state of the art on the nineties [18] mentioned that overall, there are more than 400 different types of medical robotics applications, and this number may keep increasing in the long run. On the matter of general development within rehabilitation robotics, this area has come a long way since 30 years ago the first pilot programs brought positive results but they had not commercialization plans yet. The first prototypes were limited on their flexibility and not re-programmable, but they already had a force feedback and able to repeat trajectories by demonstration through the physiotherapist.

The same review [18] addressed that in different areas there were different prototypes being tested but impossible to commercialize due to the technology and precision limitations at that time. A very important remark on the review is that in the case of assistive rehabilitation robotics, the robot arms had integrated force sensors and were guided by the operator, and the question of *"What comes next?"* was pointed towards developing more complex tasks and become the robots "intelligent" in some way. Surprisingly, there is a match between this statement and the presented use-case of ROBERT, which is a platform developed more than twenty years after the review was published. This remark shows how necessary is to look into the current situation of the platform and improve it.

A decade later, another review on the state of the art within this topic by the year 2000 [22] showed that rehabilitation robots counted with mobile prototypes with a manipulator on them, such as WALKY, RETIMO or ROMAN, which were semi autonomous.

It was also mentioned in this review the fact that rehabilitation robots started to be commercialized and the users provided useful feedback to the researchers for further improvement. The results pointed towards a better interface for non-professional user and new ideas for better safety, such as force limitation, safety strategy evaluation or robot design based on human pain tolerance [22].

Finally, in the most recent years, it is possible to narrow down the rehabilitation robots area to smaller categories, such as upper- or lower-limb rehabilitation robots, where ROBERT would

fit in.

Lower-limb rehabilitation robotics research aims to help stroke patients that have a disability due to the stroke. Projects that are related to this goal are for example: legs exoskeletons, foot orthoses or programmable foot plates [4].

Parallel to the rehabilitation robotics, there is surgical robotics within health care as well. Overall, surgical robots have a tendency to become more complex, being Machine Learning and Artificial Intelligence the last trends on the research.

Surgery robots can be divided in surgical Computer-Aided Design (CAD) and surgical assistants. In the first scenario, the robot aims to perform a procedure on the patient mostly on its own while on the second scenario the robot depends and works together with the surgeon. The second scenario can be also subdivided into surgical extenders and auxiliary surgical supports. Surgical extenders are the surgical robots that can be controlled remotely with a haptic device, which can respond to force and interact with the user [16].

The outputs on surgical extender has provided very useful outcomes in certain topics. Looking into a prototype developed for knee surgery [20] [19], the robot is manually guided by a surgeon, who receives force feedback from the robot while moving.

The following Figure 2.1 displays the 2D path that the robot has to follow on one section of the knee bone. The path is divided in three different zones and the force feedback to the surgeon may vary depending on which area the robot is currently in. As further the surgeon aims to guide the robot away from its main area (zone I), the harder will be the robot to move.

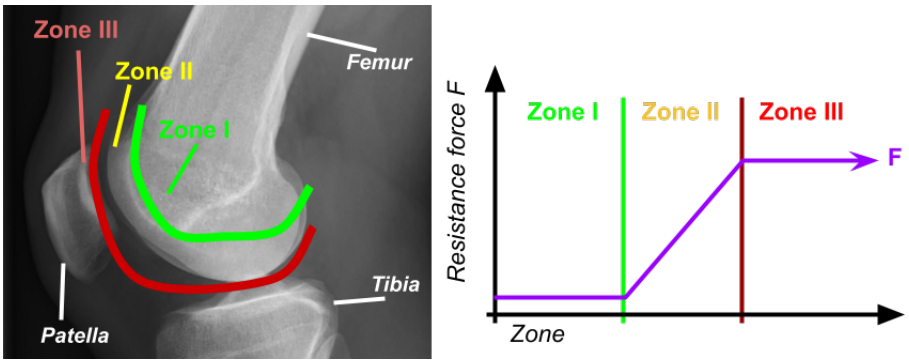

Fig. 2.1: Left: X-ray side view of the leg bones on the knee (femur, tibia, patella) and the physical constraints (red and green lines) that define the zones I, II and III. Right: resistance force in three different zones and different directions of motion constraint [19] [26].

This approach shows how to use the force resistance towards the user to prevent the robot being pushed outside an allowed area. The concept also displays how the behavior of the force is defined as a continuous and defined function within a known range (the three zones).

Path planning

Once that the robot workspace and obstacles are defined, the robot requires a path planning in order to perform a task without violating the constraints. Path planning is a wide research

area that has two main areas: global (the map and robot environment are known) and local (information about the map or environment may be missing) path planning. In both areas the procedure is the same: environmental modelling, optimization criteria and path search. On the other hand, the methods applied for this procedure may vary depending on the area.

In the case of global path planning, the most common approaches are: framework space approach, free space approach, cell decomposition approach, the topological method and the probabilistic road map method. In the case of local path planning, the most common approaches are: heuristic approach, Artificial Potential Field(s) (APF) approach, the behaviour decomposition method, the cased-based learning method and the rolling windows algorithm. Additionally, in the recent years with the new outcomes from Artificial Intelligence (AI), methods from this field such as Artificial Neural Netwrok (ANN), General Algorithm (GA) or Particle Swarm Optimization (PSO) can be used in both global and local path planning [27].

Going into more detail about the local path planning, a common heuristic method is the Dijkstra Algorithm, which directs a graph from the initial to the end point through nodes with weights, but the efficiency of this method is limited by the information about how much the robot has explored the map previously to the path planning. The other mentioned methods are the decomposition (used mostly for navigation tasks), the cased-based learning (requires a database prior to start) and the rolling windows (uses heuristic method and sub-targets to reach the goal point) that are still bringing outcomes today but they are not related to what this project aims to achieve [27].

Finally, the last mentioned approach in local path planing is using one or more Artificial Potential Field(s) (APF). This approach is widely used because of its reliability and there is a wide variety of concepts within this specific area. In general, this approach considers that the robot is a point under the influence of the field, that works as an energy field where obstacles, middle points and the target point sum or rest charges to the field. Then, the path search aims to find the optimal solution with the gradient of the field to reach the middle and target points through the map while avoiding the obstacles [17]. Depending on the concept used on the APF, the solution may be either the minimum or maximum founds on the gradient.

The approach present by Warren [25] uses an attractive artificial potential field, which means that the obstacles add positive charges to the field. The main idea is that the path to reach the goal position is the path with the lowest energy cost and least penalties. The following Figure 2.2 shows an example where given an obstacle in the middle of a field, the energy reaches its maximum value on the obstacle and it lineally decreases on the opposite direction.

Also related to this concept, there is the research presetend by Vadakkepat, Tan and Ming-Liang [23], or by Guldner and Utlun [6], which improved the gradient function definition and modified the energy cost function to find the local minimum and maximums.

Another approach for artificial potential fields was presented by Khatib [9]. In this case, the potential field is repulsive and there is a limit distance to keep from the object, where the potential energy reaches zero. The following Figure 2.3 shows an example where given a margin distance, the potential energy tends towards infinite the closer the robot moves towards the obstacle.

To summarize, both approaches [25] [9] present different but equally relevant concepts for this project to describe an APF and how to calculate and handle the potential energy generated

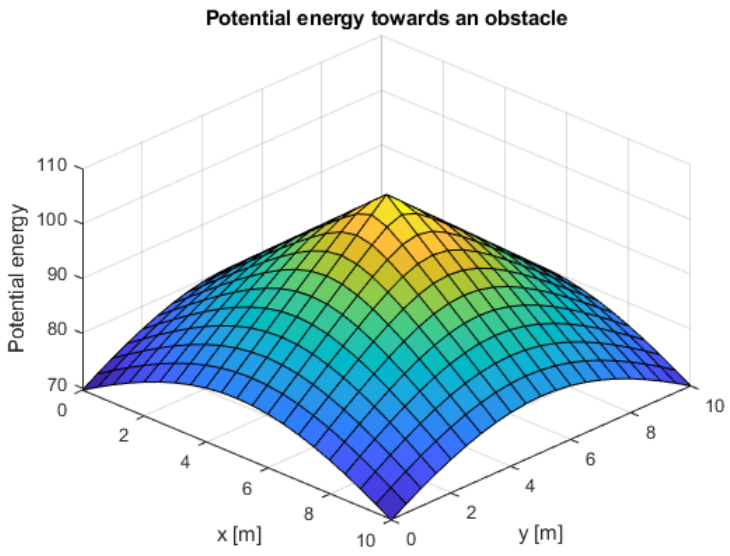

Fig. 2.2: Sample 3D potential field proposed with an obstacle on $(x, y) = (5, 5)$. Plot based on [25].

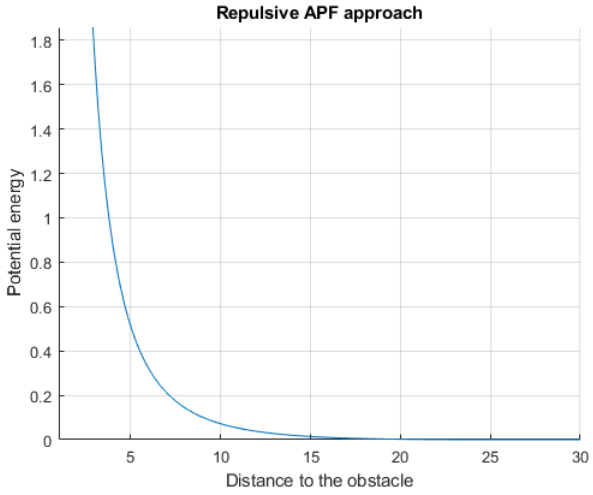

Fig. 2.3: Sample repulsive potential energy with a limit distance of $\rho_0 = 25$. Plot based on [9].

on the field.

Task motion constraints

As seen on the previous subsections, the robot shall be able to go through a planned trajectory and avoid obstacles. Then, it is also necessary to define what an obstacle is within an Artificial Potential Field(s). Using the research from Khatib [9], an obstacle and be defined as a composition of simpler primitives, such as planes, cylinders, spheres, cones, ellipsoids, lines or points.

This involves that the obstacle does not have to be a physically object, but a limitation or restriction in any other sense (e.g. force or time-variables such as velocity, acceleration) either in Cartesian space or Joint space. All in all, these constraints are considered artificial or virtual constraints and their main functionality is to add a force inside the APF.

Moving onto more recent research outputs within this topic, the paper from Hjorth et al.[7] introduces a force-position controller with virtual walls. Virtual walls are a type of artificial constraints in Cartesian space, and in this case the planes that represent physical obstacles. Additionally, the virtual walls had a spring attached to them, which is activated when the robot moves too close towards it and the spring pushes the robot back inside the allowed workspace.

The next Figure 2.4 illustrates the example use of a virtual wall with a spring that pushes a joint of the robot.:

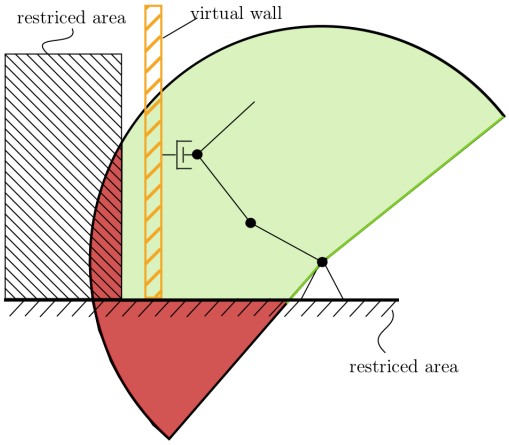

Fig. 2.4: Concept of using a virtual wall in CS. Left: actual physical obstacle. Middle: the virtual wall limits the workspace of the robot. Right: manipulator and the reachable space (green) and the collision area (red). Courtesy of [7].

2.2 General problem statement

Based on the listed disadvantages and possible system improvements in the presented Use case, the main problem of this project can be summarized as follows:

How to restrict the workspace of a robot to follow a path but still have an adaptive trajectory based on the patient's feedback or changes on the environment?

2.3 Requirement specifications

The previous sections have explained that approaches have been used within research for similar problems as stated in the Use case, and based on the problem statement, it is necessary to define a list of goals and requirements, or Research and Development Objective (RDO) for this project.

As an overview, the main goal is to develop a concept and methods for following a trajectory inside a constrained and changing workspace, which is adaptive towards the workspace and still allows the user to interact with it. The solution shall imply that the robot cannot get out of the workspace but still keep the original planned trajectory as much as the workspace allows.

Taking the main idea into account, the following Research and Development Objectives are described:

RDO1

Develop a method to adapt the trajectory to different events: workspace restrictions or the physiotherapist might want to modify it.

The proposed approach should be robust and detect in a reliable way when the patient is pushing.

RDO2

The workspace setting should be into the form of virtual constraints and take into account the given path and at least one basic constraint type in Cartesian space: a patient's bed or a wall.

The constraints might have the propiety that change overtime.

RDO3

Develop a concept to set a workspace for the robot which helps the patient to follow an exercise, but also prevent the robot to violate a Cartesian space restriction. The robot may be pushed towards the obstacle and shall stay within the allowed workspace.

The proposed approach should be able to keep the robot within the allowed workspace at least 75% of the time.

3 Solution

This chapter shall describe in general the strategy used for this project. Taking into account the related works and problem formulation, a new concept is designed. This concept is divided in different parts, and the main approach and goals of each part will also be explained within the Chapter.

First of all, based on the control scheme from [19], the following Figure 3.1 is the proposed control strategy for this project:

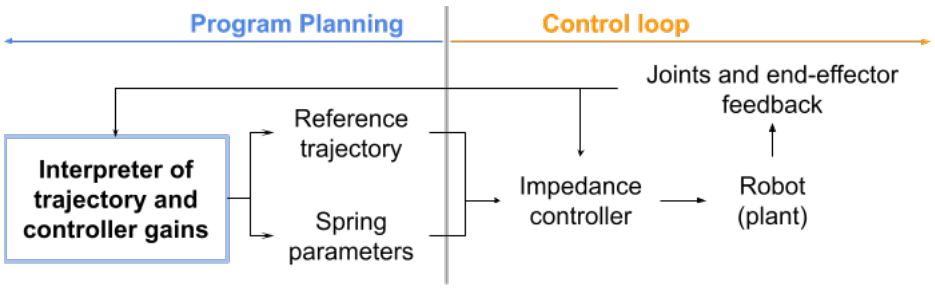

Fig. 3.1: Collaborative force-position schematic.

On the right, the control loop is the control law and plant model. In this case, the manipulator is a collaborative robot, therefore it has both the space-related feedback (position, velocity, acceleration) and also force feedback (joints torques and at the end-effector). The control loop of the schematic is explained in Section 3.1.

On the left, the program planning is the focus of this project, specifically on developing the interpreter of trajectory and controller gains. The interpreter provides the reference inputs for the controller and it is explained in Section 3.2, and subdivided in three main parts: trajectory, virtual constraints and the proposed approach.

Overall, the following sections in detail the logic and approach used for this solution from a theoretical point of view, and the concrete details and formulas are further described in Chapter 4.

3.1 Control loop

Before going into details about the controller, it is necessary to describe the plant that is controlled. As described in the Use case, the ROBERT platform has a Kuka LBR Med manipulator

with seven DOF and a linkage attached to it.

This can be represented as a kinematic chain, where all the joints are located in a frame, they are linked to each other and every joint adds at least one degree of freedom. In this case, all the joints are revolution joints, which means that one joint is equivalent to one degree of freedom and there are seven joints.

The axis frames distribution is shown in Figure 3.2. The Z axis of every frame is also the joint rotation axis. Additionally, the end-effector frame can be also added to the kinematic chain as a translational transformation.

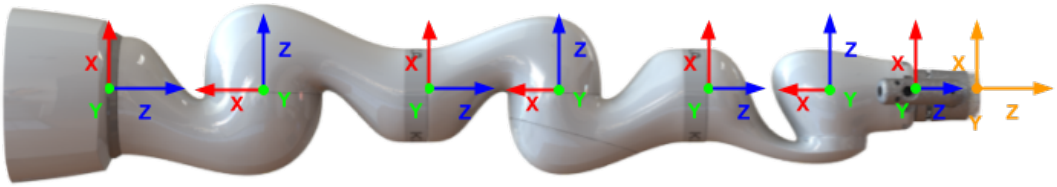

Fig. 3.2: Joint frames distribution on a Kuka LBR Med (colored frames) and the TCP frame added to the end-effector (orange).

Going into more detail into the kinematic chain, the relation between one frame and the next one can be mathematically described as a transformation matrix between the two frames.

Therefore, a kinematic chain is mathematically represented as a matrix product from the start link i to the end link j, where $i, j \in [0, n]$, being n the amount of links of the chain [3]. The following equations represent the complete transformation for the Kuka LBR Med manipulator:

$$\mathrm{T}^0_{n_\theta}(\theta) = \prod_{i=0}^{n_\theta} \mathrm{T}^i_{i+1}(\theta_i) = \prod_{i=0}^{n_\theta} \begin{bmatrix} \mathrm{R}^i_{i+1}(\theta_i) & \mathrm{P}^i_{i+1} \\ 0 & 1 \end{bmatrix} \tag{3.1}$$

where $n_\theta = 7$ is the amount n of joints specifically for the Kuka LBR Med, $\mathrm{T}^i_{i+1} \in SE(3)$ is the transformation matrix from the link i to the next one $i+1$, θ_i is the revolution joint associated with that link, and every transformation matrix contains a rotation matrix $\mathrm{R}^i_{i+1} \in SO(3)$ and a translation vector $\mathrm{P}^i_{i+1} \in \mathbb{R}^3$.

Overall, the kinematic chain describes the transformation between Joint space (JS) and Cartesian space (CS) [21]. Then, the kinematics can be subdivided in Forward Kinematics (FK) and Inverse Kinematics (IK). Forward Kinematics is the transformation from JS to CS. Given the known values of the joints position, it can be found the Cartesian position of any of the elements of the chain.

On the other hand, Inverse Kinematics is the transformation from CS to JS. Given a goal position, it can be found the joint values to move the robot to the goal. In manipulators this is usually a problem to consider due to the redundancy of the system. Luckily, there are different Inverse Kinematics solvers already available for the Kuka LBR Med, which is explained on Chapter 5.

However, the kinematics describes a space transformation and it does not depend on time. In order to have a full control, it is also necessary to take into account the system dynamics, such

as joints velocity, acceleration and forces.

The following equation shows the total forces present on a manipulator [10]:

$$\tau = M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + G(\theta) \tag{3.2}$$

where τ is the torques $n_\theta \times 1$ in the joints, M is the inertial force $n_\theta \times n_\theta$ matrix, which is the sum of all the individual inertial forces of each link and it depends on the joints acceleration. Then, C is the Colioris forces $n_\theta \times n_\theta$ matrix, given by the joints position and velocity. Both matrices M and C can be calculated using the analytical and geometrical Jacobians. Finally, G is the gravitational force $n_\theta \times 1$ matrix, given by the joints position.

Once the system modelling is clear, a control law can be implemented. As described in Figure 3.1, the control loop is the part of the control strategy scheme, which takes care of it.

There are different types of controllers for manipulators. In this case, the chosen controller is an impedance controller [8]. An impedance controller results in a robot behaving like a spring and the Hooke's law: it tends to stay in its reference position, but it can be pushed back and forth and the result force is bigger the further is from its reference point. The movement speed and brake depend on the stiffness and damping parameters on the system.

In order to reach such behavior, the controller takes into account both the position and force references and feedback in Cartesian space. Then, based on Hooke's and the previous dynamic equations, the control law:

$$F = K(x_0 - x) \rightarrow \tau = J_a^T(\theta)K(X_0 - L(\theta)), \tag{3.3}$$

where J_a^T is the transposed analytical Jacobian [10], X_0 the desired position and $L(\theta)$ is the current position.

3.2 Program planning

Going into more detail in the interpreter block, and taking into account the concept presented on [7], a new design is proposed in Figure 3.3. The block is defined as a subsystem, divided into three main parts: trajectory generation, virtual contraints and Artificial Potential Field(s) (APF).

The program planning is focused on the trajectory interpreter, which is divided in three parts: energy, virtual constraint and trajectory-related. The following diagram in Figure 3.3 shows in detail the trajectory interpreter as a subsystem with multiple blocks and which block belongs to which part.

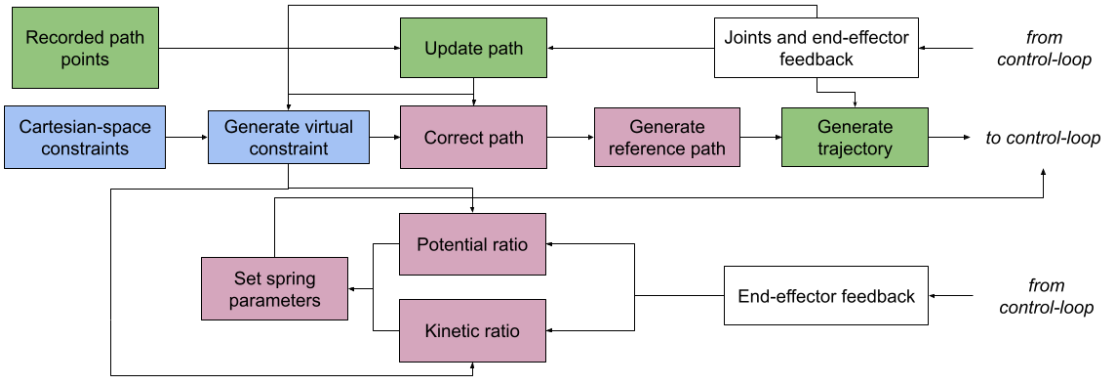

Fig. 3.3: Main block diagram of the system. Virtual constraint blocks have their backgrounds on blue, energy blocks on bordeaux, and trajectory blocks on green.

1. Trajectory blocks: they generate a path, velocity, acceleration and jerk for the robot to follow. Eventually, the trajectory could be updated overtime, so it fits new settings. The approach and details of the trajectory blocks are presented in Sections 3.2.1 and 4.1.

2. Virtual constraint blocks: they take into account the obstacles and the constrains in Cartesian space.

   These blocks check the location and forms of the present virtual constraints that the robot must avoid and notifies the closest obstacle.

   The goal of the virtual constraint is to set a consistent workspace for the robot in Cartesian space that allows the APF and trajectory blocks to find a solution. The approach and details of the virtual constraint blocks are presented in Sections 3.2.2 and 4.2.

3. APF blocks: they define the Artificial Potential Field(s) (APF), evaluates the potential present on the system based on the given reference and current position of the end-effector.

   The goal is to prevent that the robot moves too close or through a virtual constraint. The approach and details of the energy blocks are presented in Sections 3.2.3 and 4.3.

3.2.1 Trajectory generation

First of all, it is necessary to set a trajectory for the robot to follow. Taking into account the previously mentioned disadvantages of the system from the Use case, a feature to provide is to have a set of re-recorded paths and that they can be updated by the therapist.

In this project, it has been considered that the most practical option is to give the user the choice to either record a new path or load a pre-recorded one.

In the case of recording a new path for the exercise, it is assumed that it shall be recorded in the same set up as it will be used right afterwards during the exercise. Thus, does not need to be updated.

On the other hand, if a recorded path is loaded it can happen that compared to the time when the path was recorded, the robot platform has been moved or that the patient is another subject. Then, it would be necessary to update the path, so it adjusts to the current situation.

In order to adjust it, the robot activates its impedance controller and a relative slow speed and moves through the path. The impedance controller has the stiffness enough to only compensate gravity, so it is not hard to move for the physiotherapist. Then, the physiotherapist can move the robot to specific points on the run. If the robot detects the displacement, it shall replace the current goal point with the current position.

The difference between the path correction and recording a complete new path is that during a recording, the demonstration creates a new path without previous information and it depends entirely on the hand guidance while on the path correction the robot aims to follow the recorded path trajectory and in case the therapist moves it during a part of the trajectory, only that part shall be updated and the rest of the path is kept.

The following Figure 3.4 illustrates the idea of how to update the path:

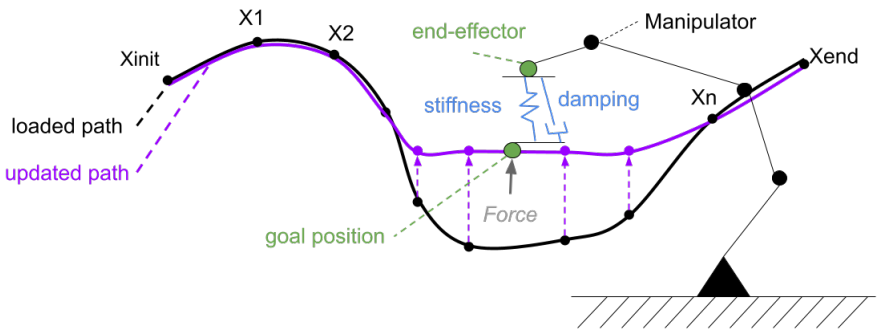

Fig. 3.4: Pre-recorded path points ($x_{init}, x_1, x_2, ...x_n$ and x_{end}), complete interpolated path (black line) and updated path with modified points (purle line and points). The stiffness and damping displayed at the end-effector are the main components of the impedance controller.

After setting a trajectory, the patient and the robot must do an exercise by completing a certain number of sets and repetitions from beginning to end. At this stage the path becomes a trajectory, and therefore the time-dependant variables, such as velocity, acceleration and jerk must be taken into account.

However, in this type of exercises, the goal of the exercise is that the patient completes the trajectory on his own, yet the robot tends to return to its goal position due to the spring behavior

of the impedance controller. This means that if the robot does not update its goal position, the robot shall return to the start point the moment the patient stops pushing. In order to avoid that, the goal point shall be updated over time while the patient is pushing. Thus, the velocity and acceleration must be defined as functions that depend on the force feedback. In this case, the velocity and acceleration work as updates for the goal position of the robot, but still let the patient to complete the exercise on his own.

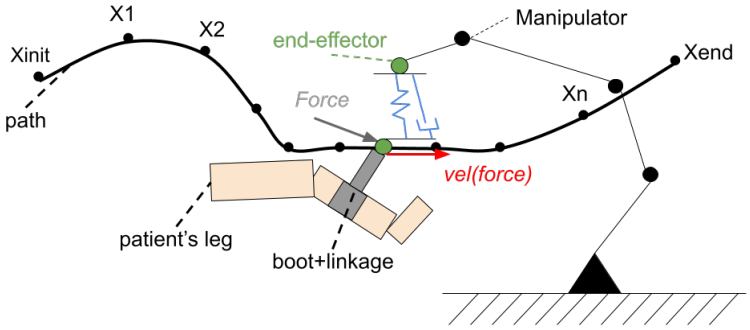

Fig. 3.5: Example exercise path on the run with a patient pushing the robot through it.

3.2.2 Cartesian-space constraints

First, it is necessary to define a workspace for the robot to operate and also for the patient to move or push the robot around. An intuitive way is to use the exercise path that the patient has to do. The therapist demonstrates it to the robot, and this will be referred as ”recorded path” through this document.

Then, the main idea is to wrap a virtual constraint around this recorded path. The virtual constraint has the shape of a tube, where the center line is the recorded path and the inside of the tube is the allowed workspace. Based on [9] and [7], virtual constraints have also a margin, which activates a repulsive force to keep the robot away, but in this case the margin is the minimum distance that the robot is allowed to keep from the constraint.

In case of loading a prerecorded trajectory, or even after updating a prerecorded path, there is still the possibility that some points or segments outside the virtual constraint and it must be corrected before the exercise starts. In case a path point or segment stays outside the virtual constraint, it must be moved back inside the allowed workspace, as seen on the scenario 2 in Figure 3.6.

Therefore, the virtual constraint must be updated on the run and consider the obstacles or constraints present inside the tube space, as seen in Figure 3.6.

3.2.3 Proposed approach

To verify if a point is inside the allowed workspace, an Artificial Potential Field(s) (APF) approach is used. Based on the previously concept described in Section 2.1, in an Artificial Potential Field(s) all the objects add a charge to the field. In the case of [9] and [7], the repulsive charge present on the field is energy measured in Joules. However, in this specific project, the

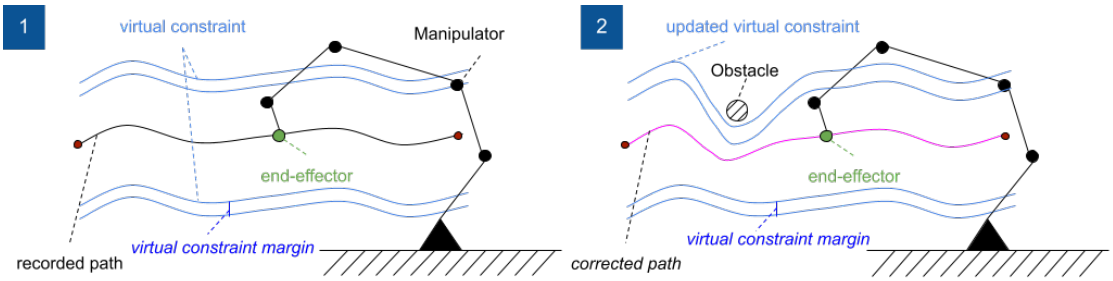

Fig. 3.6: Virtual constraint concept. 1: standard tube-virtual constraint. 2: updated virtual constraint and path after an obstacle was found inside the safe workspace.

charge is also repulsive, but also a non-dimensional unitary ratio (referred as λ from now on), which means that it does not have a unit and the minimum value is 0 and the maximum is 1.

An example of this is shown in the scenario 1 in Figure 3.7, where the ratio is zero in the center of the tube and one on the virtual wall.

After the path has been corrected and set, the same approach is used during the execution of the recorded path, to check whether the robot is being displaced from the path that is supposed to follow. In this case, the center is the corrected path, where λ equals the minimum value, and from the path line towards the updated virtual constraint, the ratio λ increases, as seen on the scenario 2 of Figure 3.7.

Finally, to avoid the robot physically being pushed through a virtual constraint, λ must be transformed from a non-dimensional ratio to a resistance force.

Taking into account that λ is a proportion that represents the distance or displacement from the goal position towards the constraint, it can be converted by lineal methods into an exponential function, and the result of such function is shown in Figure 3.8.

On the same way, another non-dimensional ratio is introduced: β. This ratio represents the proportion between the current and maximum velocity of the end-effector. The ratio β is applied on the damping coefficient of the spring, and it is converted from a non-dimensional unit into a damping coefficient through the same methods as with λ.

The main difference between λ and β is that λ depends on three different positions (the goal, end-effector and virtual constraint) while β only on the velocity of the end-effector.

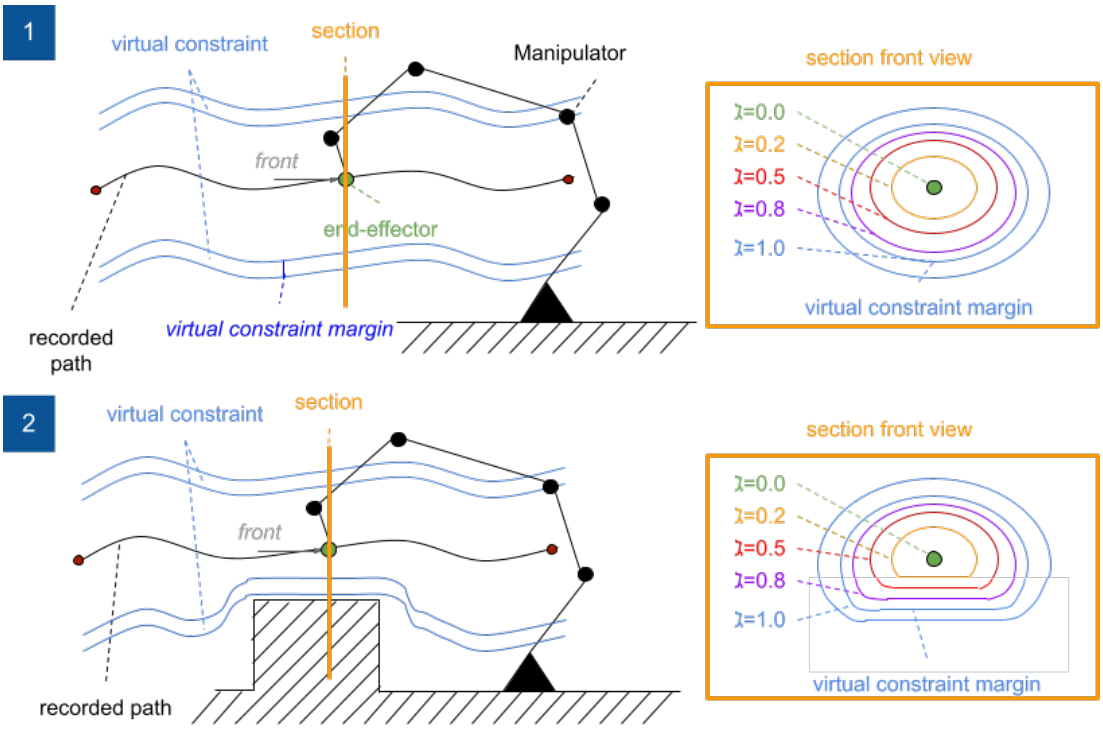

Fig. 3.7: Artificial potential field in a section of the workspace. 1: section cut and view in a goal point of the recorded path within a tube-shaped workspace. 2: same section cut and path point, but given another virtual constraint.

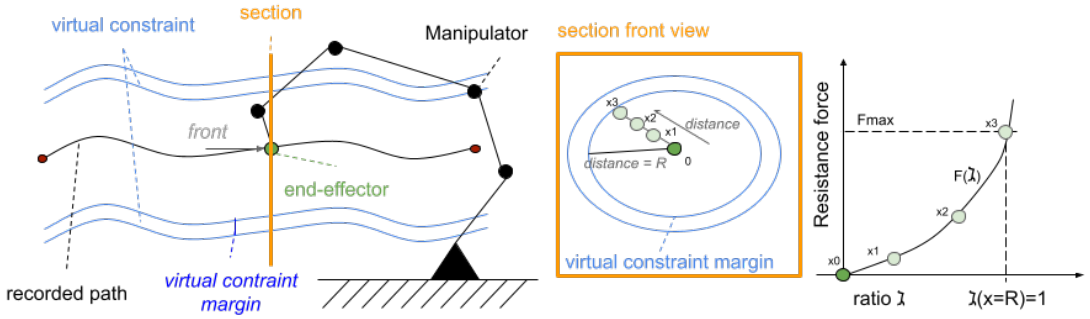

Fig. 3.8: Force giving a certain distance to the virtual constraint. Positions x0, x1, x2, and x3: arbitrary possible displacements from the end-effector, from ideally on its goal position to be close to the virtual constraint.

4 Methods

This chapter is a continuation from the Chapter 3 and it describes the proposed solution from a practical and mathematical point of view.

As a summary, the previous chapter introduced the system blocks diagram and divided into three main parts: trajectory, virtual constraint and Artificial Potential Field(s) (APF) blocks.

In this chapter, the first Section 4.1 covers the trajectory blocks, their functionalities and what they provide to the rest of the system. Then, Section 4.2 is about the virtual constraint blocks, that takes into account the present virtual constraints in the robot environment, either as physical obstacles or as workspace restrictions. Finally, in Section 4.3 the APF blocks are explained in detail. This section also describes further the proposed approach and how it affects the behavior of the robot.

4.1 Trajectory generation

This section aims to describe in detail how a hand-guided path is corrected and followed overtime using the robot feedback. As previously mentioned in Chapter 3, the specific features and functions from this section are not meant to apport a new approach in research, but rather provide a new practical use of the robot.

Based in the Figure 3.3 from Chapter 3, the trajectory blocks can be divided into two functionalities: loading a path (see Figure 4.1) and setting a path in to a trajectory (see Figure 4.2).

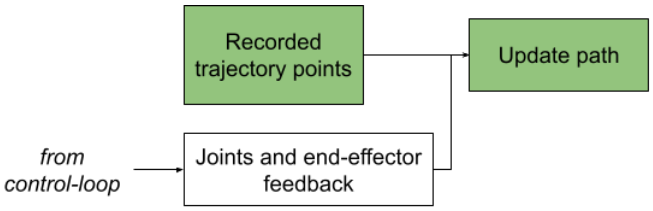

Fig. 4.1: Trajectory blocks for the loading path function. This also includes recording manually a new path.Full system block diagram on 3.3.

4.1.1 Setting a path

The first function of the trajectory blocks is shown in Figure 4.1 and it records or loads an updated path that the robot has to follow. This can be done through three approaches:

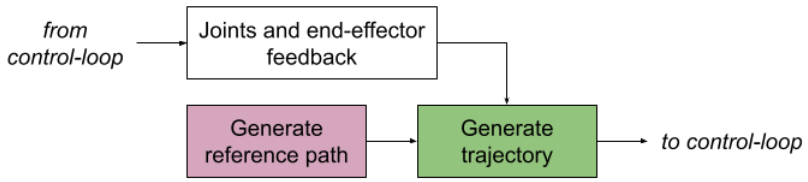

Fig. 4.2: Trajectory blocks for the following path function.Full system block diagram on 3.3.

1. Recording a new path: in the case that the therapist wants to do an exercise that is not available among the pre-recorded options, then the therapist has the option of demonstrating a path, which the robot shall use afterwards and save for future uses.

   This function uses the hand-guiding feature of the collaborative manipulator to let the therapist move the robot. At the moment in ROBERT this functionality is used differently on the guided and active mode. In the first case, the joints position, velocity and torques are recorded every $100ms$ and replicated in the same way afterwards. In the case of active mode, only the start and end points in CS are recorded. The manipulator stays on the start point and the patient has to push it far enough until it is close enough to the end point, which is usually that the robot has been pushed at least 80% of the distance between the start and end points.

   The path recording in this project is inspired on the current guided mode. For this project, it has been decided that due to the redundancy of the robot, a position in Cartesian space (CS) has multiple solutions for the joints configuration; thus, the recording process is in Joint space (JS).

   The recording process is also an accumulative process, which means that the path starts empty and it add new points the more the therapist moves the robot. However, the difference compared to the current recording process is that only the path is recorded and new path points are added based on changes of the robot position and not periodically. During the recording process, the condition to add a new waypoint is that in JS at least one joint has a position increment of $\Delta\theta \geq \Delta\theta_{lim}$, being $\Delta\theta_{lim}$ a threshold value to determine that the robot has been displaced enough.

   Finally, when the user finishes the hand-guiding, the recorded path is printed on a file, which can be used any other time in case of repeating the exercise without having to record it again.

2. Load path: as an alternative to the previous point, a new feature has been developed to load a path that has been previously recorded on the previous point.

   Then at the beginning of the exercise, the interface may ask the user if he wants to record or load a path. In case of loading, the controller creates an empty path, opens a recording-file and checks that the format is correct in order to add the containing points from the file to the path.

3. Update path: this is a feature that allows the therapist to correct the current recorded

path. This feature might be needed by the therapist in case he makes a mistake during the recording and does not want to completely record it again.

In this scenario, the robot goes through the path points with a constant joint velocity of $\dot{\theta}_{update}$. Parallel to that, the impedance controller in the robot is set with the minimum values for the stiffness and damping coefficients.

Then, when the therapist can push the robot while it is moving. The robot checks its current joint position when it finishes moving to the next path point. Then, if it is displaced from where it should be, it replaces the path point with the current position. Afterwards, it moves to the next point.

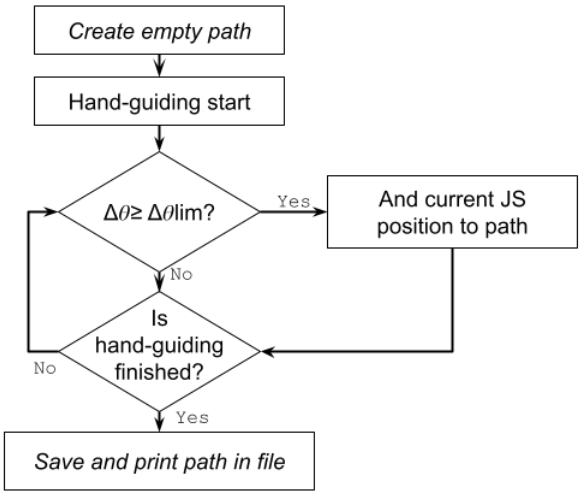

Fig. 4.3: Algorithm for the path recording using hand-guiding.

4.1.2 Following a trajectory

The other functionality of the trajectory blocks is to generate a trajectory from the path and robot feedback, as shown in Figure 4.2.

Currently in ROBERT, there are two main modes: guided and active mode. As mentioned in the previous subsection, in the guided mode the robot moves the leg through the path, using the recorded motion as the reference for the position and velocity, which means the velocity and other time-dependant variables values are fixed and they cannot be modified on the run. On the other hand, the active mode only takes into account the start and end points and it is expected that the patient pushes the robot from the start towards the end, therefore technically there is no motion in this type of exercise.

Therefore, in this project it has been added the velocity generator, which shall add the velocity to turn the path into a trajectory and its result is a variable value.

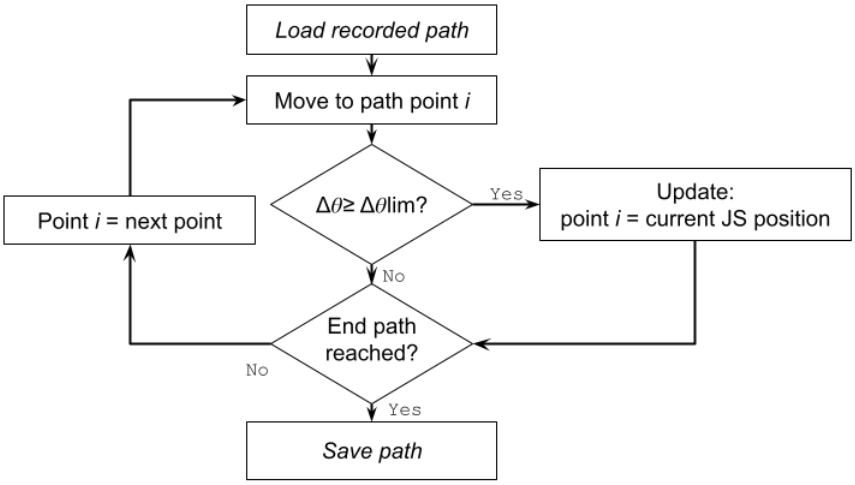

Fig. 4.4: Algorithm for the path update.

As described in Figure 3.5, in Section 3.2, the patient pushes the robot with his leg attached to the linkage of the robot. Thus, the velocity value shall be based on the force feedback from the end-effector.

Based on the presented approach from Warren [25] for Artificial Potential Field(s), that relates the relation from position and potential energy towards an obstacle in a lineal manner. In a similar way, this feature sets the velocity as a function of the force feedback, but with the difference that on the testing the force feedback value turned to be a little bit unstable and a minimum force value was added, so the original lineal equation from [25] was changed to a exponential function:

$$\dot{x}_r(f_{ee}) = me^{f_{ee}-f_{min}} + n \begin{cases} f_{ee} = f_{min} \text{ if } f_{ee} < f_{min} \\ f_{ee} = f_{max} \text{ if } f_{ee} > f_{max} \\ \dot{x}_r \in [\dot{x}_{Rmin}, \dot{x}_{Rmax}] \end{cases} \tag{4.1}$$

where $\dot{x}_r$ is the trajectory velocity in CS, m and n are the coefficient and constant values of a lineal equation, $[\dot{x}_{Rmin}, \dot{x}_{Rmax}]$ is the range value where the velocity is defined, f_{max} and f_{min} are the maximum and minimum values of the force feedback f_{ee}, which is the module of the force vector in CS: $f_{ee} =\mid (f_x, \vec{f_y}, f_z) \mid$.

Then, to fully define the function there are two conditions:

$$\dot{x}_{Rmin} = \dot{x}_r(f_{ee} \leq f_{min}) = me^0 + n$$

$$\dot{x}_{Rmax} = \dot{x}_r(f_{ee} \geq f_{max}) = me^{f_{max}-f_{min}} + n$$

Finally, the values of m and n can be calculated as follows:

$$m = \frac{\dot{x}_{Rmax} - \dot{x}_{Rmin}}{e^{f_{max} - f_{min}} - 1}$$

$$n = \dot{x}_{Rmin} - \frac{\dot{x}_{Rmax} - \dot{x}_{Rmin}}{e^{f_{max} - f_{min}} - 1}$$

The following Figure 4.5 illustrates the described approach with example values:

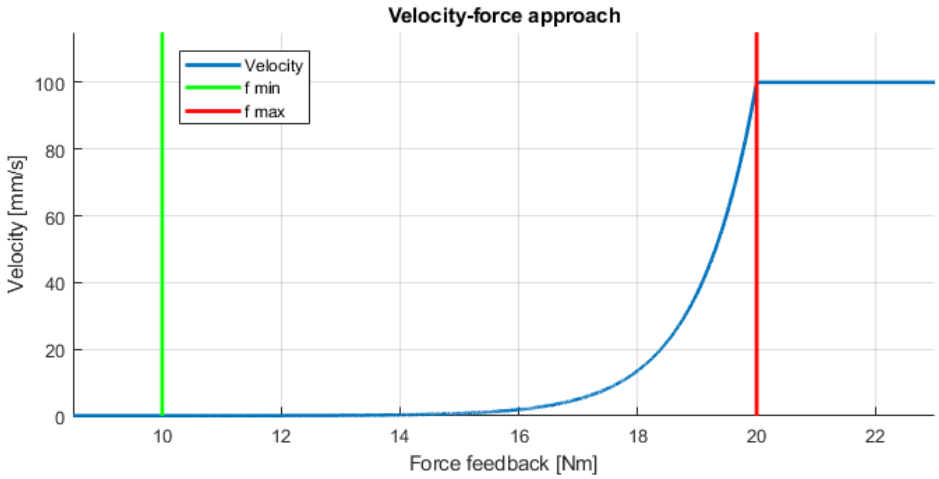

Fig. 4.5: Reference velocity value based on the force feedback vector module, with $\dot{x}_{Rmin} = 0$, $\dot{x}_{Rmax} = 100$, $f_{max} = 10$ and $f_{min} = 20$

4.2 Virtual Contraints

This section describes in detail how the workspace restrictions are set. This is a single functionality divided in two steps: setting a virtual constraint and selecting the closest point. Depending on the case, these blocks can provide information either for the path correction, as shown in Figure 4.6 or for the ratios on the run, as shown in Figure 4.7 (see next Section 4.3)

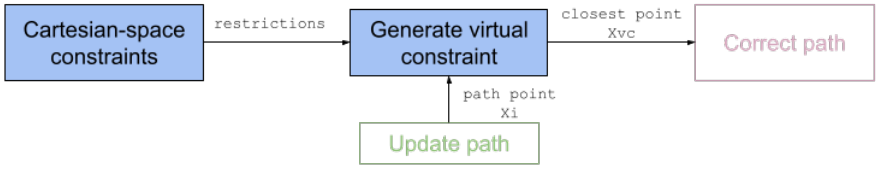

Fig. 4.6: Virtual constraint blocks for path correction. Full system block diagram on 3.3.

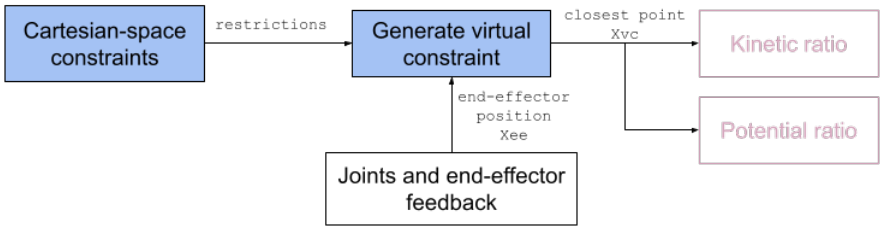

Fig. 4.7: Virtual constraint blocks for ratios. Full system block diagram on 3.3.

The next subsections describe each of these blocks, firstly how to define the virtual constrains and then how it is calculated the output from the blocks.

4.2.1 Defining constraints

Constraints are the robot motion limitations, which can be either in space, velocity, acceleration, force or in any other sense that it affects the robot motion. In case of space limitations, it can be either in Joint space (JS) (e.g. to prevent the manipulator links crashing into each other) or Cartesian space (CS) (e.g. physical obstacles).

As mentioned previously in Section 3.2, the goal is that the patient can do an exercise within an allowed space, which means that the patient can only push the robot following a given path and the robot should not deviate far from that.

Therefore, virtual constraints are introduced. Virtual constraints are, under a practical perspective, very similar to physical constraints. The main difference with a physical constraint can be represented as a simpler or as the sum of simpler virtual constraints, such as planes, points, or ellipsoids [9].

For example, as seen in the research of Hjorth et al [7] in Figure 2.4, there is a physical and a virtual plane present on the workspace of the robot. However, the virtual plane is closer to the robot and it is the limitation taken into account for the robot controller. In reality, the virtual plane does not exist and the robot can keep a distance from the physical wall.

The aim of this project, as mentioned in Section 3.2, is that the patient does an exercise and in order to do that, he has to push the robot through a given path and he cannot deviate far from that. Therefore, a virtual constraint is introduced, which is a tube in CS, whose center line is the path to follow.

A simple virtual tube is set along an axis, called projection axis and every single path point x_n can be contained within a normal plane to the projection axis. Then, the tube can be *sliced* in 2D circles, which are defined by the tube radius r_{vc} and the path point is the center. Finally, all the circle points coordinates x_{vc} within a tube slice can be defined with an angle α

The next Figure 4.8 illustrates this concept with a virtual tube example. In this example, the projection axis is the Cartesian Y axis, and the normal planes are XZ planes, where the path points $x_{init}, x_1, ..., x_n$ (being a point x a 3×1 vector in CS) can be contained separately. For a path point with the coordinates $\vec{x_n} = (x_n, y_n, z_n)$ there is a circle where a point is defined with the coordinates:

$$\vec{x_{vc}} = (x_n + r_{vc}cos\alpha, y_n, z_n + r_{vc}sin\alpha) \tag{4.2}$$

where x_{vc} is the point of the virtual point, r_{vc} is the tube radius and α the angle between the circle center (x_n, z_n) and the tube point within the normal plane.

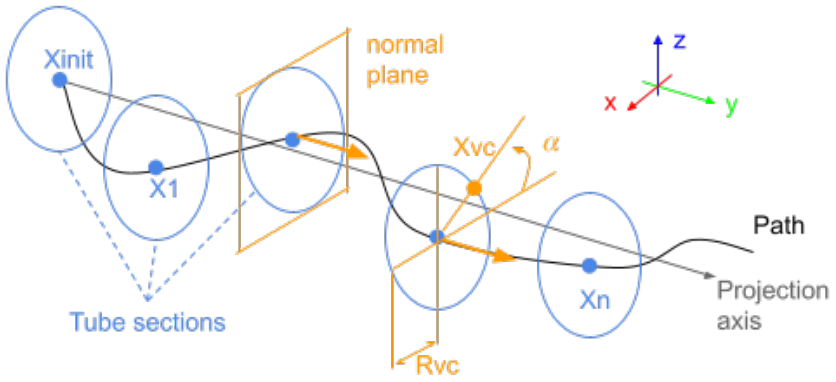

Fig. 4.8: Virtual tube example: the projection axis (grey line) is the CS Y axis, a sample 3D path (black line), with the $x_{init}, x_1, ..x_n$ points (blue points) that are the center of the tube circle sections (blue circles). The tube has a radius R_{vc} and the tube points (x_{vc}) are defined with an angle α (orange).

In the case of physical obstacles, such as the patient's bed, the floor or a wall, the easier approach is the previously mentioned [9], where this type of obstacles are represented with simpler virtual constraints.

4.2.2 Closest point of a virtual constraint

Overall, it is not practical to evaluate all the points of all the present virtual constraints. Therefore the system evaluates what is the closest point of the closest virtual constraint and the result 3D point x_{vc} is the input for the Artificial Potential Field(s) (APF) blocks, either for path correction (Figure 4.6 or for the ratios on the run (Figure 4.7).

In the first scenario, the final path that the robot must follow is still not set, so there is no virtual tube yet. Therefore, the virtual constraint blocks return the closes virtual constrain point of physical obstacles.

In the second scenario, the path to follow has been already set and a virtual tube can be wrapped around it. Then, the closest point can be either from the virtual tube or from any other obstacle.

4.3 Artificial Potential Field

This section describes in detail how an Artificial Potential Field(s) (APF) approach is used and how it affects the trajectory and the controller behavior. As an overview, there are two phases of the main block diagram from Figure 3.3 where the APF approach is used: on correcting the path and on setting the spring parameters.

Overall, the main idea behind the APF is have a limited and defined value to correct the recorded path and change the stiffness and damping of the robot at the same time. Thus, one of the most important characteristics of a non-dimensional APF approach is that the field is repulsive and mapped with a known range of values in any case, it does not matter the actual dimensions of the workspace or how far a reference point is from a virtual constraint. Another aspect of a non-dimensional APF is that it can be extrapolated to different uses, as the stiffness and damping, which is explained in Section 4.3.3.

In the case of the path generation, the Figure 4.9 illustrates the blocks where the APF is used to correct the recorded path according to the present virtual constraint and its output is the CS path reference for the trajectory generation. The detailed method and output are explained in Section 4.3.1.

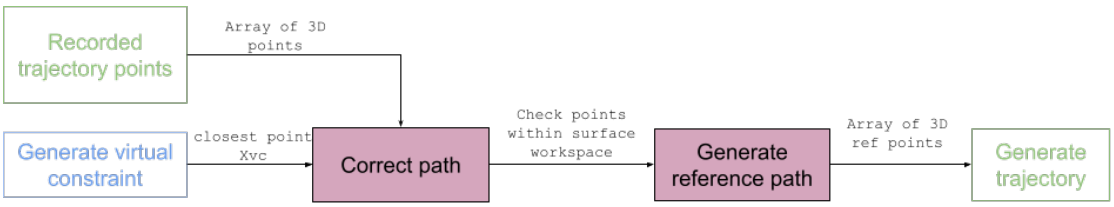

Fig. 4.9: APF approach block diagram for path correction. Full system block diagram on 3.3.

Then, the spring parameter set-up is shown in Figure 4.10. In this case the inputs are the APF approach outputs are the ratios λ and β (see Section 3.2.3), and the inputs are the goal position, the closest point of the virtual constraint and current position and velocity of the end-effector. The detail equations and description of both ratios are explained in Section 4.3.2.

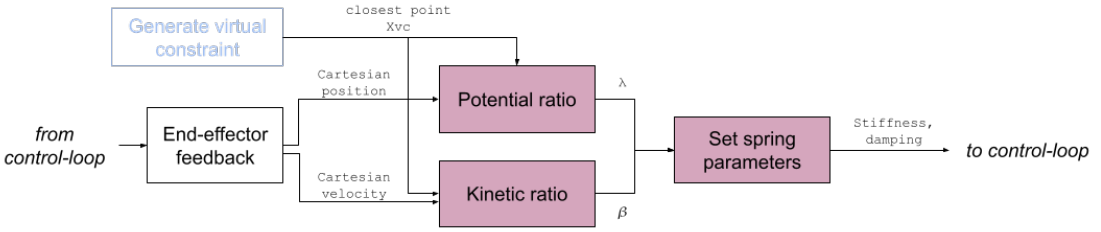

Fig. 4.10: APF approach block diagram for path following. Full system block diagram on 3.3.

Afterwards, the ratios λ and β are used for the spring parameters set-up and the results are forwarded to the controller, which is described in 4.3.3.

4.3.1 Path correction

Before the robot starts following a trajectory, the recorded trajectory must be checked and modified if needed. Even if a path has been recently recorded and/or updated as described in Section 4.1, changes in the environment might happen, and an obstacle might be on the way.

Based on [9], the ratio λ for all the path points x_i is calculated using the following equation 4.3.1:

$$\lambda(\rho) = \frac{1}{2}\eta\left(\frac{1}{\rho} - \frac{1}{\rho_0}\right)^2, \in [\lambda_{min}, \lambda_{max}] \tag{4.3}$$

where ρ is the distance from the centroid to the current point and ρ_0 is the maximum possible distance, which is from the centroid to the virtual wall; and η is a numerical value that defines the speed of the exponential function.

The same equation can be re-written as:

$$\lambda(x_i) = \frac{1}{2}\eta\left(\frac{1}{\mid \vec{x_{vc} - x_i} \mid} - \frac{1}{\mid \vec{x_{vc} - x_{center}} \mid}\right)^2, \in [\lambda_{min}, \lambda_{max}] \tag{4.4}$$

where x_i is the path point i as a 3×1 vector in CS, and with the same format there is x_{center}, which is the centroid location and it indicates which side of the virtual constraint is allowed, and x_{vc} which is the closest point from the virtual constraint.

In order to keep the APF flexible, a non-dimensional margin δ is introduced, which is the proportional distance from 0 to 1 that the robot must keep from the virtual constraint, being 0 that the robot can reach the virtual constraint and 1 that the robot cannot move from the center. Therefore, the ratio λ increases until the robot reaches the maximum allowed distance $\delta\rho_0$, where it reaches the maximum value.

Then the value of η is calculated as follows:

$$\lambda_{max}(\rho = \delta\rho_0) = 1 = \frac{1}{2}\eta\left(\frac{1}{\delta\rho} - \frac{1}{\rho_0}\right)^2$$

$$\eta = \frac{2\delta^2\rho_0^2}{\delta^2 - 2\delta + 1} \tag{4.5}$$

The following Figure 4.11 shows an example application which illustrates the concept. In the figure it is possible to see how λ increases exponentially towards the virtual constraint location and how δ affects the closer it is possible to get towards it.

Path points that are outside the allowed workspace due to being too close to the virtual wall return a λ value outside the range $[\lambda_{min}, \lambda_{max}$, such as $\lambda >> \lambda_{max}$,. In order to correct the path, points that are not within the range, are pushed back inside the allowed workspace, by setting them to the closest allowed towards the workspace.

However, the λ takes into account how close a point is towards a virtual constraint in proportion to the maximum distance possible and this has a flaw on one single situation: on points

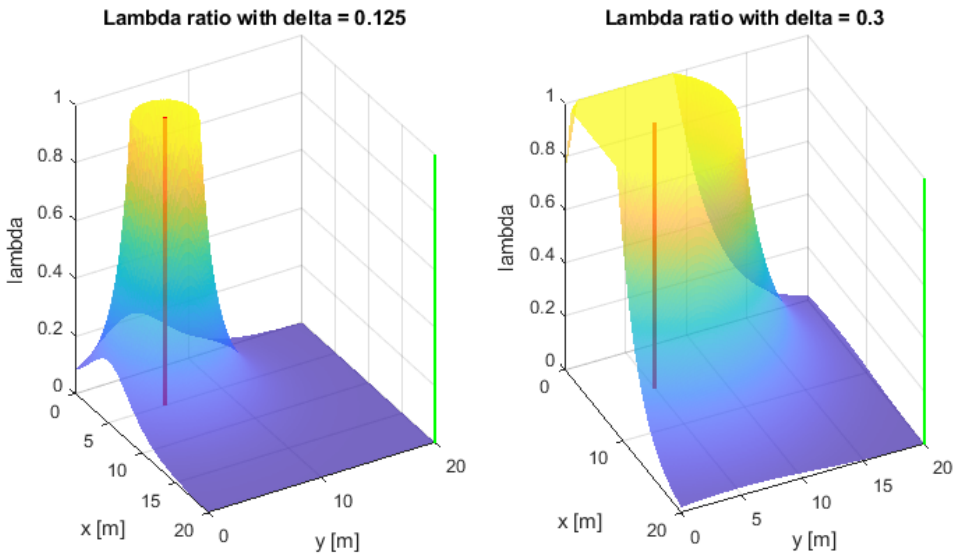

Fig. 4.11: Example of an APF with $x_{center} = (20, 20)m$ (vertical green line) and $x_{vc} = (5, 5)m$ (vertical red line) for two different margins values $\delta = 0.125, 0.3$.

that have an acceptable distance towards a virtual constraint but they are on the other side of the virtual constraint. Therefore a new element is presented:

$$\rho_1 = \mid \vec{x_{center}} - x_i \mid \tag{4.6}$$

Finally, to fully define the APF, there are three conditions:

$$\lambda(\rho = \rho_0) = 0 = \lambda_{min} \rightarrow \lambda(x_i = x_{center}) \tag{4.7}$$

$$\lambda_{max}(\rho) = 1 = \lambda(\rho = \delta\rho_0), \delta \in (0, 1) \tag{4.8}$$

$$\begin{cases} \lambda(x_i) \in [0, 1] & \text{if } \rho \geq \delta\rho_0 \text{ and } \rho_1 <= \rho_0 \\ \lambda(x_i) = 1 & \text{if } \rho_1 > \rho_0 \end{cases} \tag{4.9}$$

The following Figure 4.12 shows a given path with waypoints and the correction to fit the path within an allowed workspace and a margin distance.

4.3.2 Ratios on the run

Once a corrected path has been set, the robot aims that the patient follows it, but the patient can still push it away on the run. Thus, the APF is also used during the runtime. In this scenario

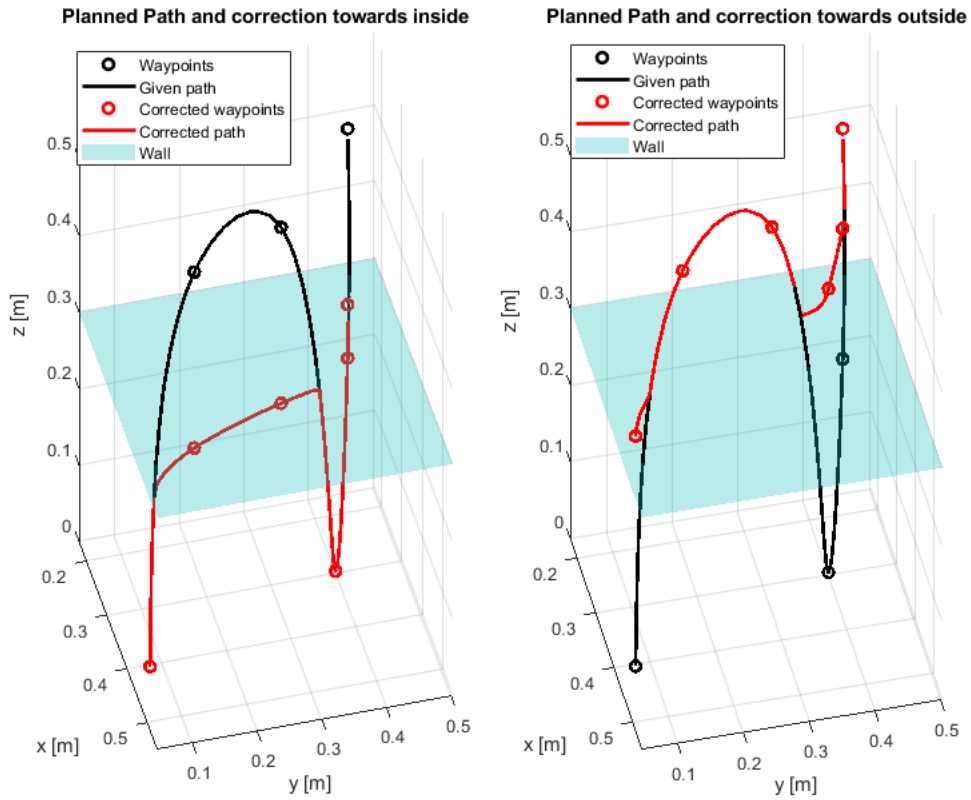

Fig. 4.12: Example of a path (black lines) correction (red lines) using the APF approach and a virtual wall (blue plane). The allowed workspace is either under the virtual wall (plot on the left) or above it (plot on the right).

there are two different ratios: λ and β.

The first ratio, λ is used as described on the equation 4.3.1, with the differences that the ratio is applied on the position of the end effector $x_i = x_{ee}$ and the centroid location is on the goal position $x_{center} = x_{goal}$. Due to this, the minimum value λ_{min} is located on the path that the robot aims to follow.

A detail to take into account is that in the theoretical scenario that the robot could overcome the margin distance towards a virtual constraint, then $\lambda \to \infty$ according to the equation 4.3.1, which means that there is no maximum value defined for λ. Therefore the following condition is added:

$$\lambda(\rho < \delta\rho_0) = \lambda_{max} = 1$$

The following Figure 4.13 displays a sample scenario, where three sample virtual constraints have been included: two points and a wall. The figure aims to illustrate the behavior of the APF when the robot moves from the reference position towards any virtual constraint in space.

However, this example is not realistic because it takes multiple virtual constraints into account instead of a single point.

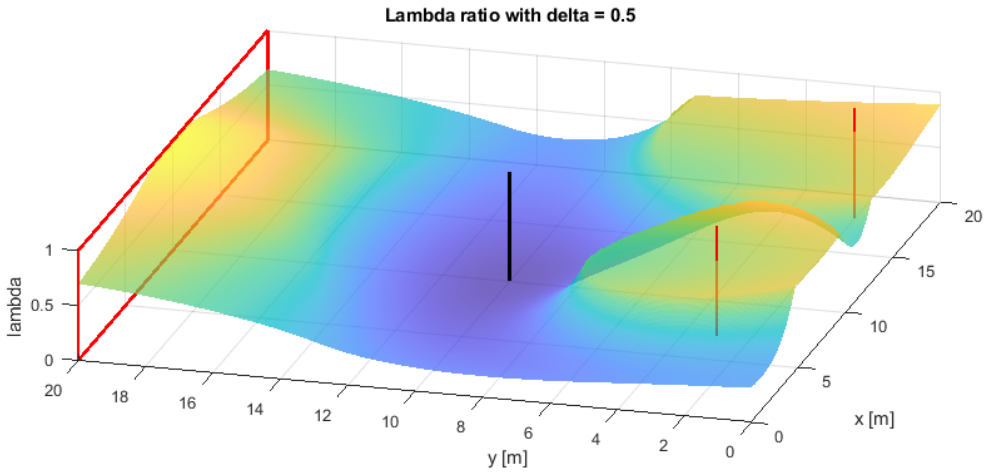

Fig. 4.13: APF with center on $x_{center} = (10, 10)$ (black line), two constraint points in $x_{vc1} = (7, 3)$ and $x_{vc2} = (18, 1)$ (red lines) and a virtual wall on $x_{wall} = (x, y = 20)$ (marked with a red rectangle).

The second ratio, β is used to prevent the robot reaching the maximum velocity. Time-dependant variables must also be taken into account in motion planning and collaborative environment. Then, based on Equation 4.3.1, β can be defined as:

$$\beta(\dot{x}_i) = \frac{1}{2}\eta_\beta \left(\frac{1}{\mid \vec{x_{max}} - \vec{\dot{x}_i} \mid} - \frac{1}{\mid \vec{x_{max}} \mid} \right)^2, \in [\lambda_{min}, \lambda_{max}] \tag{4.10}$$

where $\dot{x}_i$ is the current velocity in CS, $\dot{x}_{max}$ is the maximum velocity allowed in the system, and η_β is a constant which can be calculated in the same manner as in Equation 4.3.1, by setting also a non-dimensional limitation $\delta_\beta \in (0, 1)$ and replacing rho_0 by $\dot{x}_{max}$.

Finally, to fully define β within a range:

$$\beta_{min} = 0 = \beta(\dot{x}_i = \dot{x}_{min} = 0) \tag{4.11}$$

$$\beta_{max} = 1 = \beta(\dot{x}_i = \delta\dot{x}_{max}) \tag{4.12}$$

where the limit values of the function $\beta_{min} = 0$ and $\beta_{min} = 1$, and the limit values of the velocity $\dot{x}_{min} = 0$ and $\dot{x}_{max}$ is the maximum allowed velocity. However, the maximum velocity might have a different numerical value depending on the situation.

4.3.3 APF application

To prevent the robot being continuously pushed or make it go through a virtual constraint, the controller settings being updated overtime and increase its resistance force, as shown in Figure 3.8. In the impedance controller described in Section 3.1, the robot behaves like a spring, which has two coefficients in Cartesian space (CS): the stiffness k_c and the damping d_c.

Therefore, the final goal of the APF from the previous sections is to provide the non-dimentional ratios λ and β that can be used for setting the stiffness and the damping in Cartesian space. In case of the stiffness k_{cs}, it depends on the ratio λ and the damping d_{cs} on β.

For these coefficients, the proposed equations are a lineal transformation from non-dimensional ratio to a gain:

$$k_{cs}(\lambda(\rho)) = (k_{max} - k_{min})\lambda + k_{min} = (k_{max} - k_{min})\frac{1}{2}\eta\left(\frac{1}{\rho} - \frac{1}{\rho_0}\right)^2 + k_{min} \tag{4.13}$$

$$d_{cs}(\beta(\dot{x}_{ee})) = \frac{1}{2}\eta_\beta\left(\frac{1}{\mid \dot{\vec{x}}_{max} - \dot{\vec{x}}_{ee} \mid} - \frac{1}{\mid \dot{\vec{x}}_{max} \mid}\right)^2 \tag{4.14}$$

where $k_{max}, k_{min}, d_{max}, d_{min}$ are maximum and minimum values for the stiffness and damping coefficients, which depend on the robot and model.

The following Figure 4.14 shows an example of Equation 4.13 with $\delta = 0.4$ within a tube. This example illustrates how the stiffness increases inside the tube space, being the goal position x_{center} the minimum value.

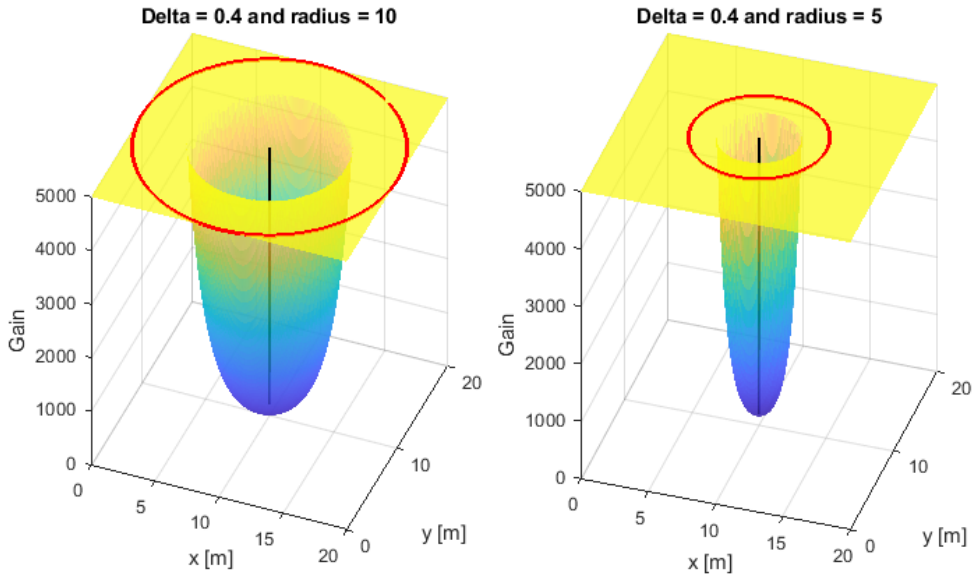

Fig. 4.14: Stiffness gain with goal position on $x_{center} = (10, 10)$ (black line), a margin $\delta = 0.4$, $k_{max} = 5000$ and $k_{min} = 200$ for two different scenarios. Left: constraint tube has a radius of $\rho_0 = 10m$ (red circle). Right: constraint tube has a radius of $\rho_0 = 5m$ (red circle)

5 Implementation

This chapter shall explain how the presented solution and methods from Chapters 3 and 4 have been programmed and used on ROBERT.

The proposed approach implementation has been divided in two main components: a Kuka LBR Med robot (in a ROBERT unit) and in CoppeliaSim (a simulation model). The following block diagram is similar to Figure 3.3 and it specifies which blocks have been implemented in which side.

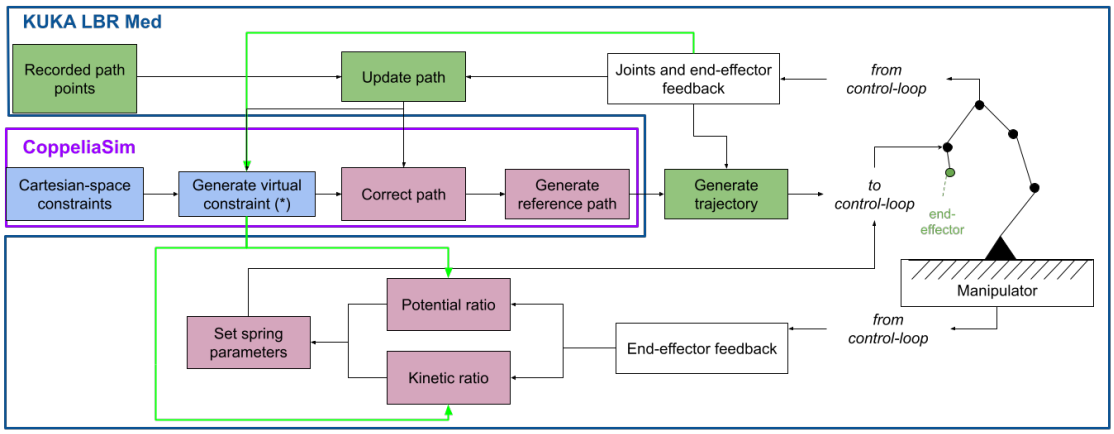

Fig. 5.1: Main block diagram of the final implementation. (*): this block has been implemented in both CoppeliaSim (for the path correction, marked in black arrows) and in the Kuka LBR Med (for the APF ratios, marked in green arrows).

After choosing which side is in charge of which blocks, it is necessary to assign what tasks are done by which side. The following list enumerates the present components in the implementation and the corresponding tasks:

1. Kuka LBR Med (ROBERT): implements the methods for λ and β on the run (see Section 4.3.2) and the impedance controller gain values application (see Section 3.1).

2. CoppeliaSim (Simulation environment): implements the virtual constraints and the ratio methods for the path correction (see Section 4.3.1).

3. Matlab (optional display): shows the values of the ratios and gains on the run from the

real ROBERT. This element is only to show and it does not implement any specific functionality.

The list can be summarized and illustrated with the following diagram in Figure 5.2, which shows the listed elements, their functions and what information is shared among them:

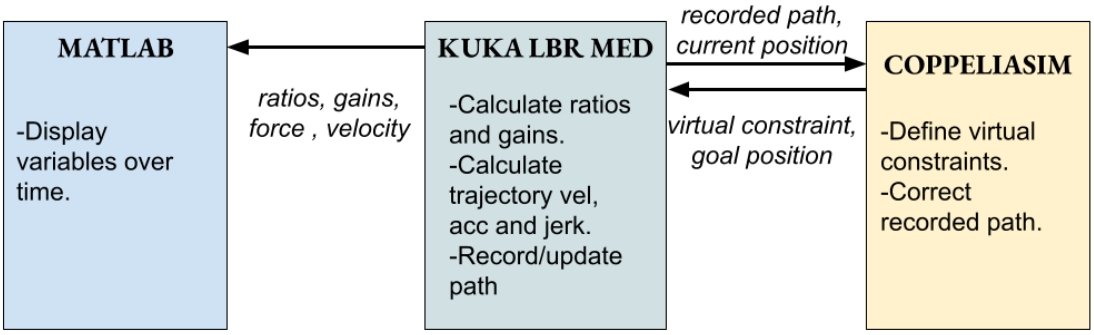

Fig. 5.2: Implementation components overview

The next sections in this chapter shall describe more in detail the robot and the simulation: Section 5.1 is about the simulation model and environment, including how the ROBERT virtual model has been developed and the objects present in the scene. Then Section 5.2 describes how the Kuka LBR Med controller has been set. In case of using Matlab as a display, it only plots the information received by the Kuka LBR Med.

The last part of this chapter is Section 5.3, which describes the final robotic application developed for the project and how the simulator and robot work together.

5.1 System simulation

This section describes the simulation used in this projects, including the environment, element and what aims to achieve.

First of all, the real robot platform does not include a vision system, nor additional sensors to detect object around it and no resources were given to improve the current situation. On top of that, even if resources were given, there is currently only one ROBERT unit available that is shared by the team of Life Science Robotics (LSR).

Therefore, a simulation environment is needed to overcome these difficulties. The chosen simulator is CoppeliaSim, where the simulation elements and robots are programmed in LUA language [2] and has a free-licensed educational version. The simulator has also additional plug-ins for importing robot models and kinematics trees. On top of that, the simulation can run dynamically with changes, which means that the objects present in the scene can be manually moved around and updated on the run.

In the matter about this project, a simulation model of ROBERT has been developed from the CAD files from the original ROBERT model and the Kuka LBR Med manipulator [14]. The

kinematics were specified on the ROBERT simulation model as described previously in Figure 3.2 and the joints limits as specified by the manufacturer [13].

The following Figure 5.3 shows the model rendered in SolidWorks 2019 and the details of the end-effector and the location of the TCP frame. Additionally, the manipulator base inclination and height on the platform has also been included.

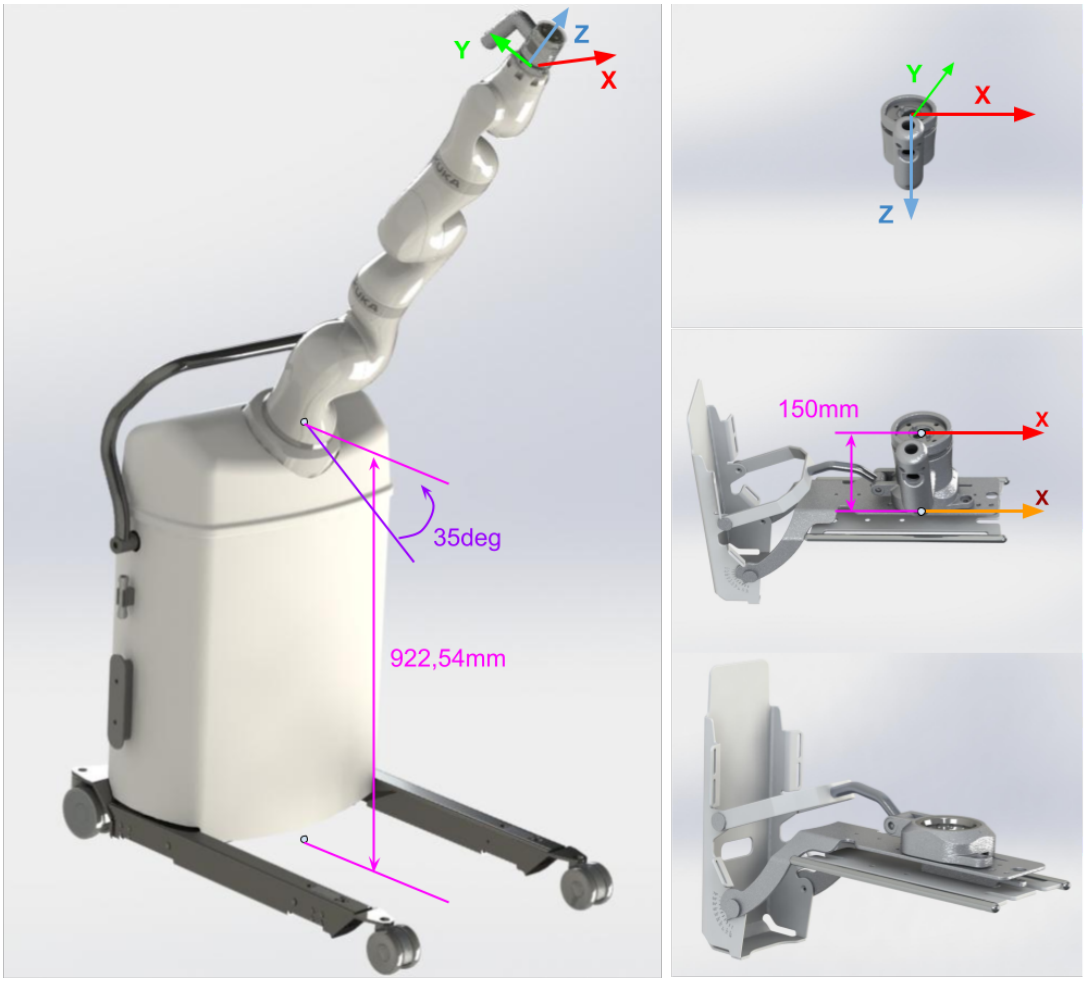

Fig. 5.3: ROBERT model in SolidWorks. Left: complete platform and the θ_7 frame location. Top right: θ_7 frame front view with only the linkage. Center right: linkage and boot attached, with the X axis from the θ_7 frame and TCP frame. Bottom right: only boot attachment.

Afterwards the model was exported from SolidWorks into CoppeliaSim. One important detail about importing the robot into the simulator is that the boot complex shape makes the physics engine on the simulator run much slower. Thus, there are three different versions of the ROBERT model in CoppeliaSim, so the simulation set up is easier to handle, as shown in Figure 5.5.

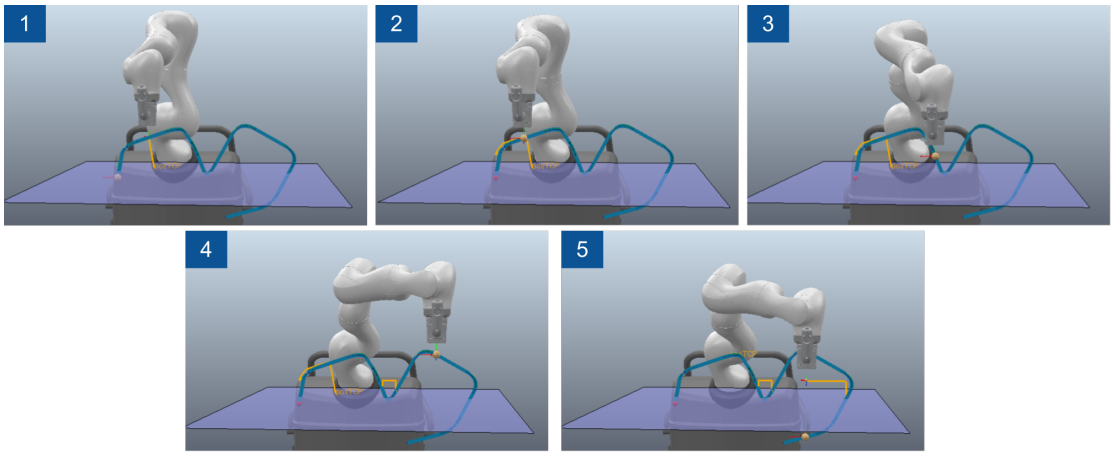

Fig. 5.4: Implementation of the path correction method with a virtual wall. In an example path sequence, the robot follows a path (3D blue line), but if the pointer goes too close or towards a wall, the goal position still remains within the allowed workspace (3D orange graph).

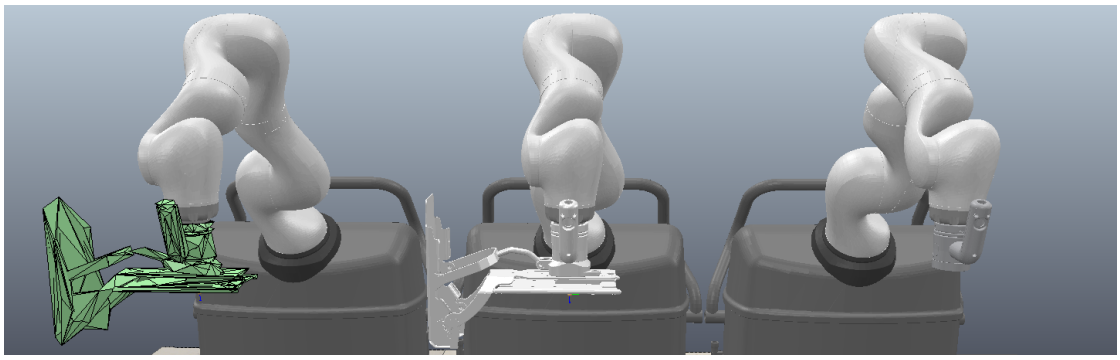

Fig. 5.5: Different end-effector versions. Left: simplified linkage and boot for more efficiency on simulation run time. Middle: linkage and boot in high details. Right: only linkage.

On the matter of details, the simulator also includes a position controller for every joint and an IK module for manipulators, so the physics engine already implements the dynamics and equations described in Section 3.1. An example of the IK is shown in Figure 5.4, where the simulation implements the path correction methods as described in Section 4.3.1 and the robot follows the corrected sample 3D path.

Additionally, an example of the manipulator dynamics is shown in Figure 5.6, where a mannequin works as a patient and it hits the robot with the lower-limbs. The figure shows how the developed ROBERT simulation model works as a respondable and dynamic object that can interact with other objects within the simulation scene.

However, only one type of controller can be active at the time, so a manipulator can only be moved by setting positions in JS through the joints control or in CS through the IK module. Thus, there are two main ROBERT models in CoppeliaSim with the same end-effector: one version which is controlled in JS and the other one in CS.

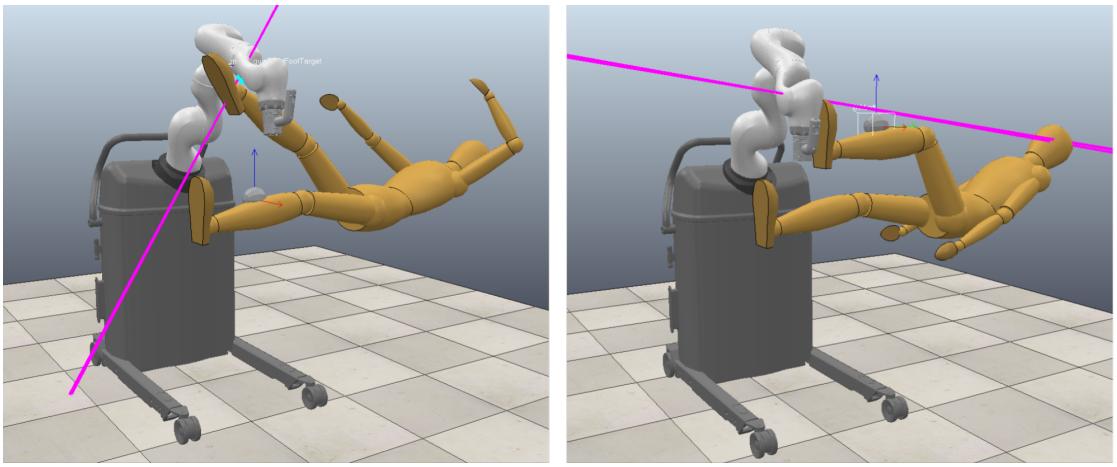

Fig. 5.6: ROBERT model in Coppelia and a mannequin. On the simulation run, the mannequin moves the legs to different positions and it hits the robot. An optional feature displays the force generated as pink strikes when two objects collide with each other.

After developing the ROBERT simulation model and its variants, the scene for the final robot application is shown in Figure and it has the following elements:

1. A virtual wall on $Z = 1052mm$, which represents the patients' bed as an obstacle.
2. A UDP socket to communicate the real and simulation ROBERT. This feature functionality was tested during the simulation model development and the Figure 5.8 shows a sample application, where the real robot follows the simulation motion through a sample 3D path.
3. Two ROBERT simulation models: one controlled in JS and the other in CS. The two robots are required to transform the path points from JS to CS and vice-versa.

The main script of the simulation scene takes care of all the present elements and it implements the path correction methods as described in Section 4.3.1.

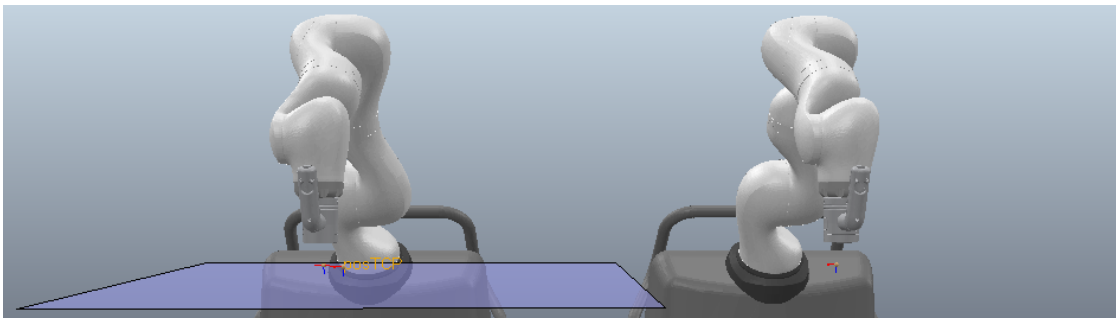

Fig. 5.7: Final scene for the implementation with a CS controlled ROBERT (left), a JS controlled ROBERT (right) and a virtual wall (transparent plane).

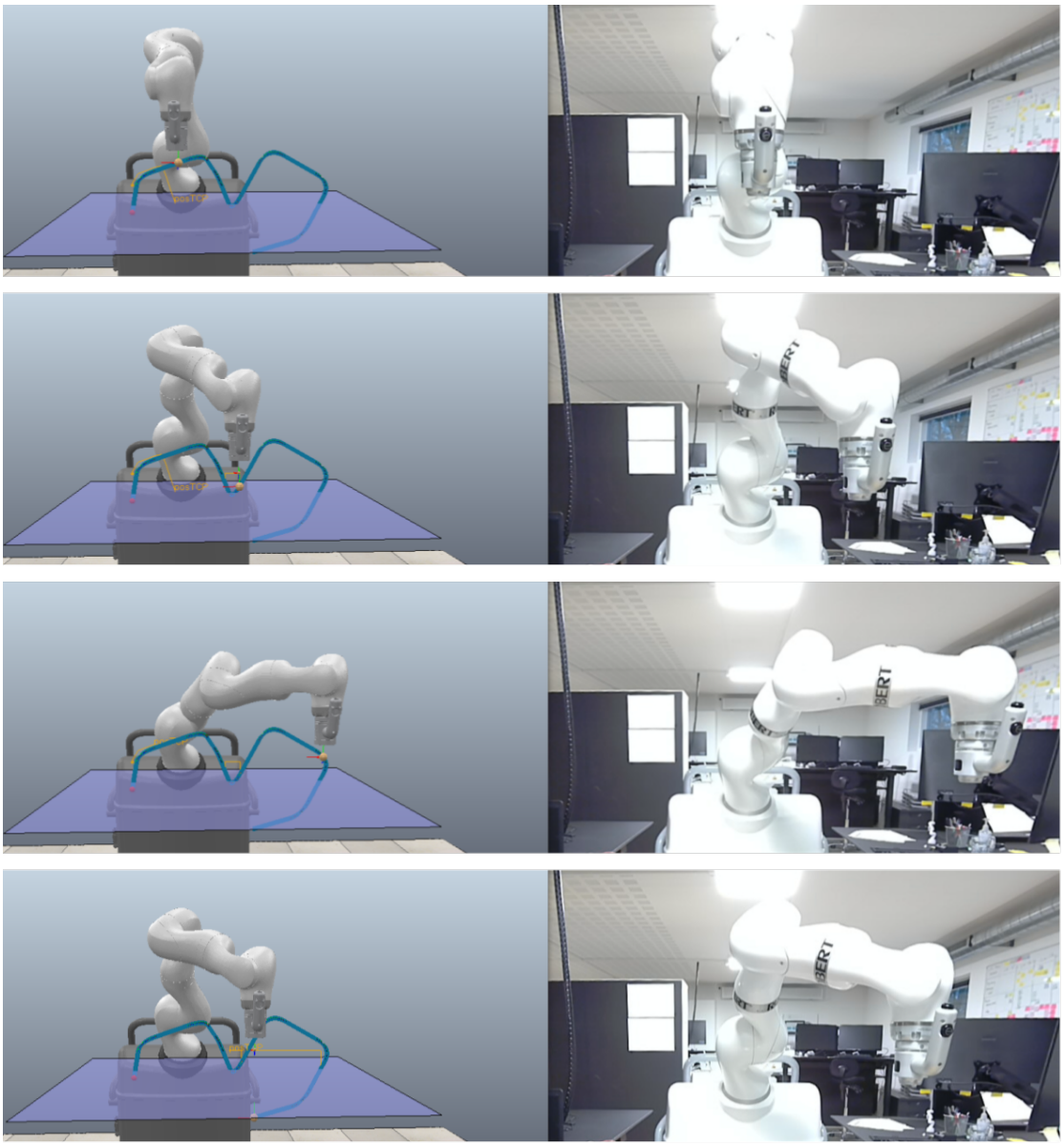

Fig. 5.8: Sample developed application to test the communication between the real and simulation ROBERT models. The ROBERT simulation (left) corrects and follows a 3D path, while the real ROBERT (right) copies the motion from CoppeliaSim.

5.2 Real system

This section describes how a physical ROBERT unit is programmed and how it has been set for the development.

In this project the implementation on a ROBERT unit has been done using the Kuka Sunrise Workbench, which is the IDE used by the Research and Development department at LSR.

The Kuka Sunrise Workbench is an IDE similar based on Eclipse: the programming language is Java and it integrates all basic Java functionalities plus a KUKA API specific for Kuka manipulators programming and control [12]. The implementation of the APF, calculating the ratios λ, β and the application of the ratios on the gain values has been done in Java. Then, downloading the program from the IDE to the controller is through the X21 Robot interface [11]. Afterwards, in order to communicate the Kuka LBR Med program with other external software, the same interface can be used for communication IP. As previously mentioned, the communication protocol with CoppeliaSim is UDP.

The following Figure 5.9 shows a picture of the robot next to the working desk, which was the main space for the testing. Due to the limited physical space, it is more practical to use the robot only with the linkage and interact with it with the hand rather than with a leg attached to the linkage through the mechanical boot.

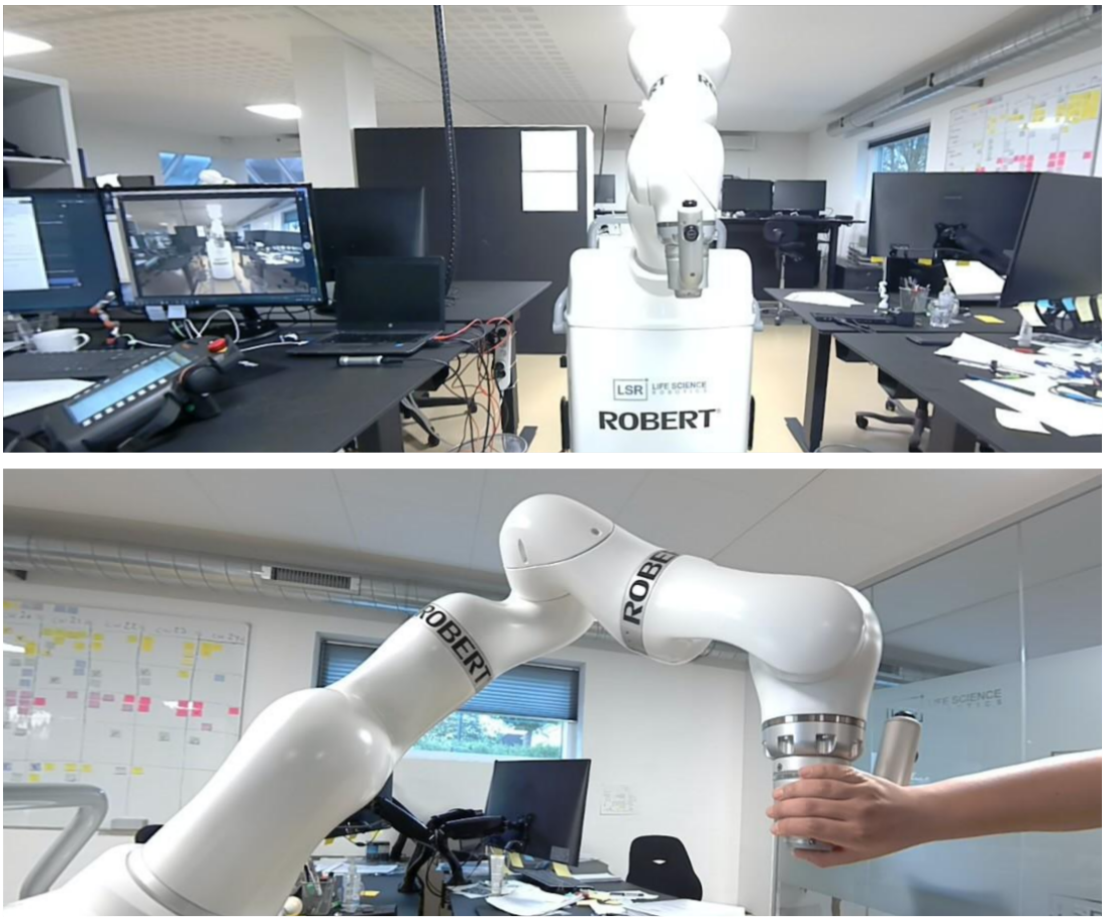

Fig. 5.9: Real ROBERT workspace. Top: ROBERT usual location for development. Bottom: interacting manually with the robot by pushing or pulling the linkage.

5.3 Final robotic application

This section describes in detail the application developed for implementing the proposed approach in this project. As an overview, the complete system follows a state machine diagram, which is on the Kuka LBR Med side. The state machine executes the process with the following steps:

1. Initial status: this status aims to start up the robot and it implements the recording and update trajectory methods described in Section 4.1.1. The user is given the option to choose either to record a new trajectory with hand guiding (see Figure 4.3) or to load a pre-recorded trajectory and update it if necessary (see Figure 4.4).

   Parallel to this, the simulation starts as well and it waits for the real robot to be finished with setting a path.

2. Waiting status: the goal of this state is that the handguided path is transferred from the real to the simulation ROBERT. The reason of this information exchange is that the simulation needs to collect the path waypoints before correcting them according to the virtual wall.

   In the mean time, the real ROBERT waits until the simulation finishes collecting all the path points.

3. Running status: the simulation and the real ROBERT move in sync through the path points. In order to achieve it:

   (a) The simulation ROBERT corrects the first path point and moves the robot there. Then the position of the simulation ROBERT is sent to the real robot among with the closest virtual constraint point from the virtual wall. For the real ROBERT the received position is x_{goal}.,

   (b) The real ROBERT receives the goal position, virtual wall closest point from the simulator, and also the current end-effector position and force feedback from the plant.

   Based on that information, the robot velocity is calculated based on the current Cartesian force applied by the user at the end effector. Afterwards, the ratios λ, β and stiffness and damping gains are calculated. In order to do so, the application chooses as the virtual constraint point x_{vc} either the information from CoppeliaSim or the closest point from the virtual tube, depending on the position of the end-effector and which of the two constraints is the closest.

   (c) In order to update the motion with new velocity and/or gain values, the current motion must be cancelled and re-started. Since it is not practical to do this action in every loop, this is only done if at least one of the parameters has an increment or decrement of: $\mid 0.1\dot{x}_{max} \mid$ in the velocity or $\mid 0.01k_{max} \mid$, $\mid 0.01d_{max} \mid$ in one of the controller gains.

   (d) While the real ROBERT moves towards the goal position, the simulator waits and checks constantly the virtual wall constraint and updates x_{goal} if necessary.

(e) When the real ROBERT reaches the position x_{goal}, it will notify the simulation ROBERT and both sides will increment their count of finished path points and the process will repeat again.

4. End status: in case the last finished point is also the last path point, the simulation will notify the end of the exercise to the real ROBERT and both sides will start from the beginning again.

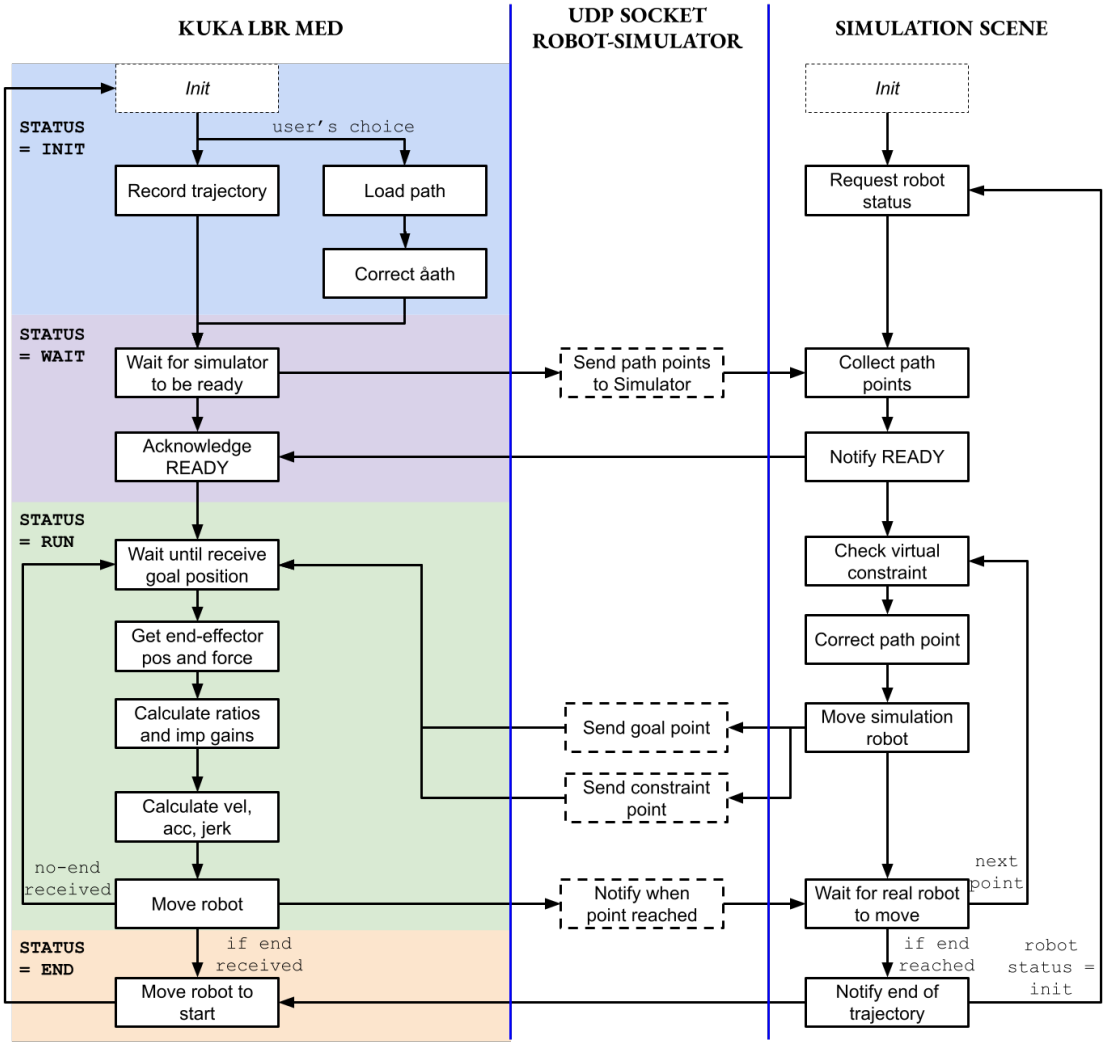

Fig. 5.10: Main robotic and simulation application diagram.

6 Test and results

This chapter describes the testing process for the proposed approach of this project, including the types of testing and the results.

First of all, the presented methods in Chapter 4 require limit and constant values in the equations, which in this case are specific for the Kuka LBR Med and the type of application. The limits and values are divided in three areas:

1. Path and trajectory generation (Section 4.1): for the algorithms described in Figures 4.3 and 4.4, and the Equation 4.1.2 the Joint space (JS) and Cartesian space (CS) velocity is a non-dimensional variable within the range $[0, 1]$, which is the $\%$ of the maximum allowed velocity, which is $250mm/s$ for the Kuka LBR Med.

   The recorded path for all the tests in this chapter is a knee-extension exercise that has been recorded using the hand-guiding algorithm described in Figure 4.3.

   All the parameter values are specified on Table 6.1.

2. Virtual constraints: the tests have been done using one type of constraint a the time, either a virtual wall or a virtual tube.

   In the case of a virtual tube, the projection axis is the Cartesian Y axis because it is the direction that the recorded path is more aligned with and three arbitrary values have been chosen for the tube radius. On the other hand, for the virtual wall, a plane on the Cartesian Z axis has chosen to simulate the constraint of the patient's bed, and the height is an arbitrary value similar to where a real patient's would actually be.

   All the parameters values to define the constraints are specified on Table 6.2.

3. Using an Artificial Potential Field(s) (APF) for keeping the robot within the allowed workspace (Section 4.3): when using the Kuka impedance controller [12], there are six different gains for the stiffness and damping in CS. The six gains are: three the Cartesian axis x, y, z and the Cartesian angles C, B, A, and the gains for the Cartesian axis are within the range of $[0, 5000]N/m$ and the gains for the Cartesian angles are within the range of $[0, 300]N/m$.

   In case of the stiffness, the limit values depend on the manipulator model. In case of the damping, the coefficient is a non-dimensional ratio within the range $[0.1, 1]$.

   The complete list of the parameter values are specified on Table 6.3.

The next sections present the results tables and the most relevant graphs of each type of tests. First, Section 6.1 describes how the trajectory generation parameter values have been set, how the method were tested and the outcomes. Then, Section 6.2 covers both the virtual constraints and APF methods, testing the path correction and the ratios with the two different types of constraints. Finally, Section 6.3 are tests that review the complete presented approach and the functionality of the developed application as a whole.

Table 6.1: Parameter values for trajectory generation

Variable	Value	Units	Comment
$\Delta\theta_{lim}$	0.075	rad	Experimental value
f_{max}	27	N	Experimental value
f_{min}	20	N	Experimental value
$\dot{x}_{Rmax}$	0.99	non-dimensional	Limit value based on specifications [12]
$\dot{x}_{Rmin}$	0.001	non-dimensional	Limit value based on specifications [12]
$\dot{\theta}_{update}$	0.2	non-dimensional	Experimental value

Table 6.2: Parameter values for virtual constraints

Variable	Value	Units	Comment
Z_{vw}	1052	mm	Virtual wall Z axis component
r_{vc}	500, 250, 125	mm	Virtual tube radius

Table 6.3: Parameter values for APF methods

Variable	Value	Units	Comment
δ (path correction)	0.01, 0.1	non-dimensional	Experimental values
δ (on the run)	0.125,0.3,0.75	non-dimensional	Experimental values
k_{max} (axis)	5000	N/m	Limit value based on specifications [12]
k_{min} (axis)	200	N/m	Experimental value to compensate gravity
k_{max} (angles)	300	N/m	Limit value based on specifications [12]
k_{min} (angles)	10	N/m	Experimental value to compensate gravity
d_{max}	1.0	non-dimensional	Limit value based on specifications [12]
d_{min}	0.1	non-dimensional	Limit value based on specifications [12]
δ_β	0.125	non-dimensional	Experimental value

6.1 Trajectory generation

This Section covers an justifies the presented values on the Table 6.1, more concretely the f_{min} and f_{max}. Additionally, all the following tests were run without a virtual constraint and without using the APF methods.

In order to determine the value of f_{min} two tests were made: the robot moves to the start point of the path and it stays still with no velocity. Then on the first test, it was recorded the static force present at the end-effector for lifting the linkage. On the second test the robot had small sporadic pushes from a person on the end effector, and it was recorded the size and length of the peaks. The second test results represents if the sporadic pushes are big enough in orded to be taken into account for setting the value of f_{min}.

Finally, on a last test, a person simulating being a patient, would manually pushed the robot through the path points. The result of this test would show the average and standard deviation values of a constant push, which would give as a result the value for f_{max}.

The following Figure 6.1 shows the force recorded at the end-effector on the three tests. In case of the first test, the static force shows a constant behavior with an average of $3.9044N$. In case of the second test, sporadic pushes are shown as peaks over time. The maximum recorded peak was had a value of $19.5204N$, which is approximately four time the average static force value. Then, as an output of the first two tests, the f_{min} was set on $20N$.

Afterwards, the third test calculates the average value of the constant push taking into account only the force values above f_{min}. The result is an average of $27.7259N$ and a standard deviation of $5.3509N$. Therefore, the value of f_{max} was set on the average value.

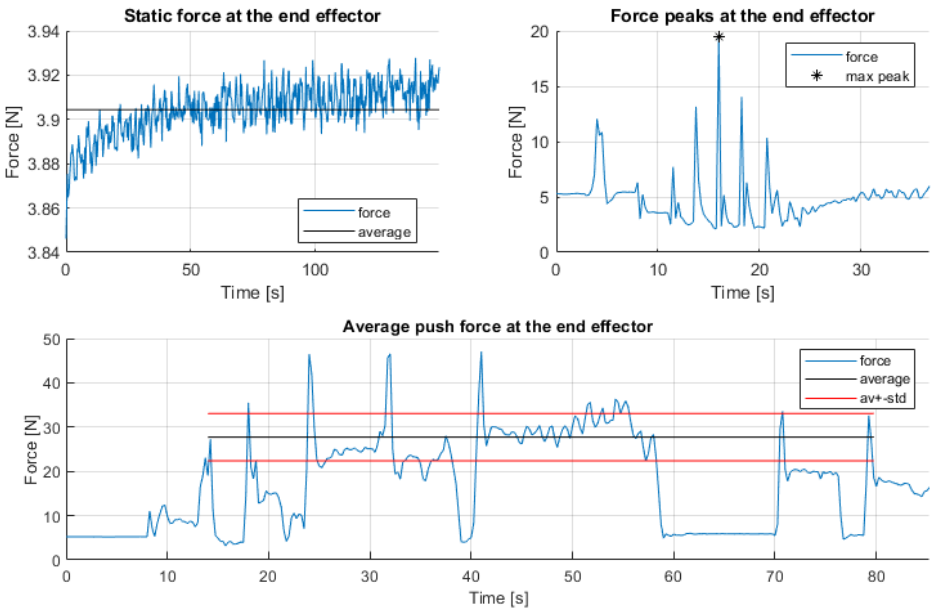

Fig. 6.1: Force recordings for the three tests to set the f_{min} and f_{max} values. Top left: static force test. Top right: small sporadic pushes test- Bottom: pushing the robot through the path.

Finally, trajectory generation approach proposed on Section 4.1 was tested by setting the f_{min} and f_{max} values in the implementation of the Equation 4.1.2 in the robot software and pushing the robot through the same path 10 times.

The Figure 6.2 shows three out of the 10 tests, that are the most representative of the outcomes. To summarize the three graphs:

- Graph 1 shows the recording nr.1. In this case, the push is mostly constant over time. However, the velocity is mostly low during until $t = 20s$ and mostly active after $t = 30s$.
- Graph 2 shows the recording nr.6. In this case, the push is not completely constant, but it has some stop-moments around $t = 30s$ and $t = 60s$. On the other hand, the velocity has a similar behavior, by being constant most of the time and close to zero in the same time intervals.
- Graph 3 shows the recording nr.7. In this case, the push is almost constant overtime and the velocity is equal to the maximum value most ot the time as well.

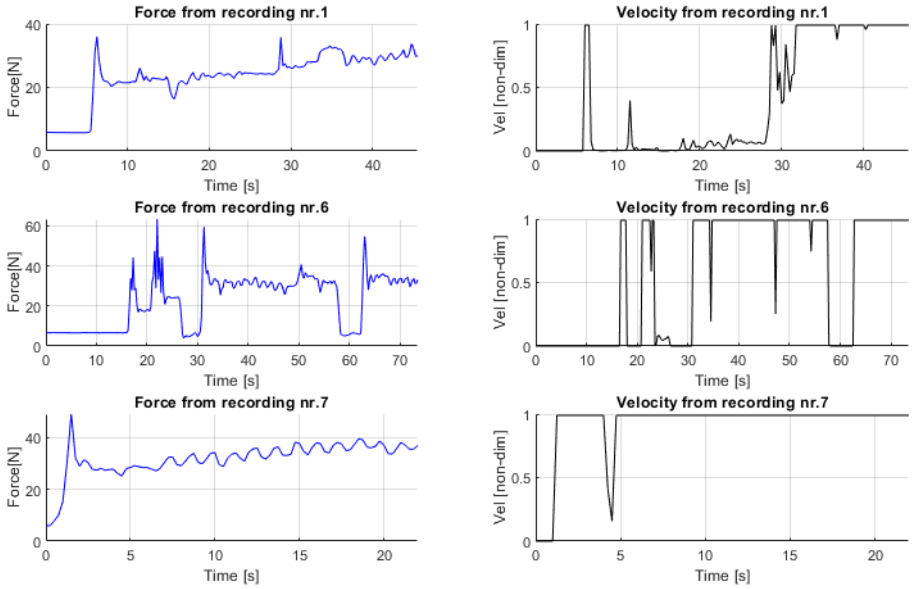

Fig. 6.2: Force-velocity recordings using Equation 4.1.2.

6.2 APF and virtual constraints

This section covers the testing for the implementation of the virtual constraints and Artificial Potential Field(s) (APF) methods. The testing in this section is purely focus on these two elements, therefore the trajectory generation functionality from the previous section is not active in these tests. Instead, a constant joint velocity of 0.2 has been set for the motion.

The APF methods and virtual constraints are partly implemented in CoppeliaSim and partly in the Kuka LBR Med and this can be confusing for the reader. Thus, this list provides an overview of how the testing was done:

- The APF application on the run with λ and β ratios are implemented on the Kuka LBR Med and the testing procedure might vary depending on the type of present virtual constraint:
  - In case of a virtual tube, this type of virtual constraint is already implemented in the robot controller.
  - In case of a virtual wall, the simulator must send to the robot the virtual wall closest point to the current end-effector position, which means that both the simulation scene and the robot controller are needed for these tests.
- The path correction is a functionality that is implemented on CoppeliaSim simulator and it has been tested together with the Kuka LBR Med when using a virtual wall.

The first virtual constraint is the virtual tube and the corresponding tests and results are detailed explained in Section 6.2.1. Secondly, the tests with a virtual wall involving the Kuka LBR Med and CoppeliaSim are described in Section 6.2.2. In both cases for the virtual tube and the virtual wall, it is the same criteria for the results evaluation and analysing the values of δ, ρ, λ.

Thirdly, the velocity ratio β has a different approach, as described previously in Section 4.3. Therefore, in this chapter, the tests and results for this parameter are described separately in Section 6.2.3.

Overall, all the tests were done with three different values for the margin $\delta = 0.125, 0.3, 0.75$ and for every margin value, 10 tests were run. During the tests, a user simulating to be a patient, pushed manually the robot around. The idea was to start with small pushes and increase them gradually and try to push the robot through the virtual constraint, either slowly pushing towards it or with a short, strong push from the goal position towards the constraint.

In case of the virtual constraints and the potential ratio λ, the results of the tests can be labeled according to the following criteria:

- Successful: the end-effector position kept a minimum distance of $\delta\rho$ from the virtual constraint.
- Acceptable: the end-effector position did not keep a minimum distance of $\delta\rho$, but it did not violate the virtual constraint.

- Failed: the end-effector went through a virtual constraint. It is considered that the robot violates it if the absolute ρ distance to the virtual constraint is lower than $10mm$.

In case of the velocity ratio β, the results can be labelled with the same categories as listed here, but the interpretation on the graphs is different, please see Section 6.2.3.

6.2.1 Potential ratio with a virtual tube

This subsection describes the tests and results for the presented APF approach with a virtual constraint with the shape of a tube, as described in Section 4.2. In this case, the tests have been run only with a tube as virtual constraint. Therefore, the center line of the tube is the path that the robot has to follow. Then, the maximum distance rho_0 is the tube radius r_{vc}, which is constant.

As mentioned previously, the tests were run with three different values for δ and three for r_{vc}. For every combination of values, 10 tests were run, making in total 90 tests.

For every recorded test there are two graphs to evaluate the result: one on the left side with $rho_0, \delta\rho 0, \rho$ and one on the right with the result λ. On the left side rho_0 (green line) and $\delta\rho 0$ (red line) are the limit values that the end-effector can keep from the virtual constraint, and rho (blue line) the distance of the end-effector towards the virtual tube, which should be between both. In case of going above the rho_0 line, it is not a problem, but going under the $\delta\rho_0$ line means that the margin distance could not be kept. On the right side the graph shows the ratio λ for every moment of rho.

The following Figure 6.3 shows three of the tests and how the result is interpreted:

- Top graphs: are the test nr.3 for the parameters $\delta = 0.3, r_{vc} = 250mm$, which is a continuous push. In this case, rho is always between rho_0 and $\delta\rho_0$, and the ratio λ is updated accordingly. Therefore, the result of this test is successful.

- Middle graphs: are the test nr.7 for the parameters $\delta = 0.75, r_{vc} = 500mm$, which is an increasing push. In this case, rho goes towards $\delta\rho_0$ and it overcomes it, but the distance towards the virtual constraint is at least $350mm$. On the right side, and the ratio λ reaches λ_{max} when the distance is on the line $\delta\rho_0$ and stays on the maximum value. Therefore, the result of this test is acceptable.

- Bottom graphs: are the test nr.1 for the parameters $\delta = 0.125, r_{vc} = 125mm$, which is a test with sporadic pushes with a continous push at the end. In this case, rho goes back and forth from the goal position towards the virtual constraint, overcoming the margin distance $\delta\rho_0$ multiple times and also reaching distance values under 10mm. Finally, during the continous push, it stays close to the $\delta\rho_0$ margin. On the right side, λ also goes back and forth from λ_{min} and $lambda_{max}$ until reaching a stable value during the continuous punch. Therefore, the result of this test is failed.

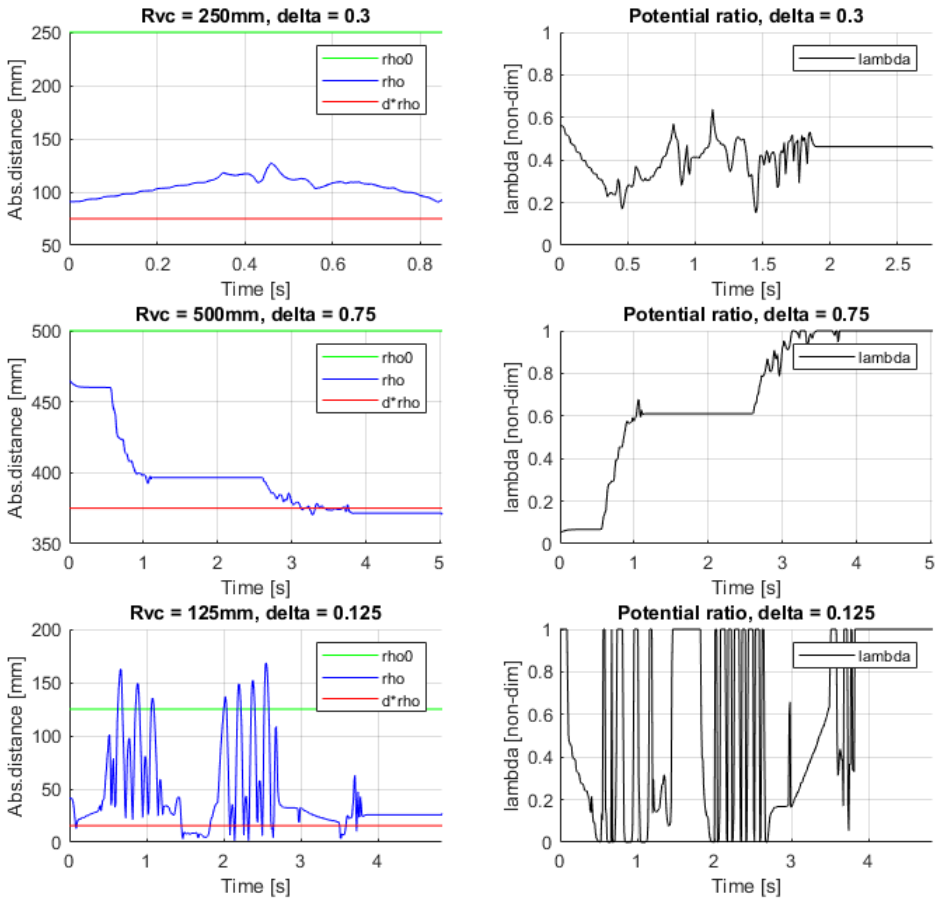

Fig. 6.3: Three test results for the potential ratio λ for a virtual tube. Left side: absolute distances and distance margins. Right: λ ratio. From top to bottom: test 3/10 for $\delta = 0.3, r_{vc} = 250mm$, test 7/10 for $\delta = 0.75, r_{vc} = 500mm$ and test 1/10 for $\delta = 0.125, r_{vc} = 125mm$.

Based on this criteria, the following Table 6.4 summarizes the results for all the tests with the different parameter value. For every row, the most repeated result is marked in a different color, being only in the case of $\delta = 0.3, r_{vc} = 125$ when two different results were given the amount of times. In this specific case, it is considered that the result "Failed" had the majority:

Table 6.4: Virtual tube test results for different δ and r_{vc} parameters.

δ (0,1)	r_{vc} [mm]	Successful	Acceptable	Failed
0.125	500	9	1	0
	250	8	2	0
	125	0	4	6
	Total =	*17 (56.67%)*	*7 (23.33%)*	*6 (20%)*
0.3	500	9	1	0
	250	3	7	0
	125	0	5	5
	Total =	*12 (40%)*	*13 (43.33%)*	*5 (16.67%)*
0.75	500	2	8	0
	250	0	9	1
	125	0	7	3
	Total =	*2 (6.67%)*	*24 (80%)*	*4 (13.33%)*

Overall, the most successful ratio is in the scenario of $\delta = 0.125$ (17 successful, 7 acceptable, 6 failed), followed by $\delta = 0.3$ (12 successful, 13 acceptable and 5 failed) and $\delta = 0.75$ (2 successful, 24 acceptable, 4 failed).

In the case of virtual tube radius, the parameter $r_{vc} = 500mm$ has the highest amount of successful results (21 successful, 9 acceptable and 0 failed), followed by $r_{vc} = 250$ (11 successful, 18 acceptable, 1 failed) and $r_{vc} = 125mm$ (0 successful, 16 acceptable, 14 failed).

6.2.2 Potential ratio with a virtual wall and path correction

In this case the two parameters that were tested were the margin δ for the path correction (applied on CoppeliaSim) and the margin δ that the robot has to keep from the virtual wall.

For this testing, the virtual wall height on $Z = 1052mm$ was too high to apply values over $\delta = 0.1$ because the corrected path points were placed outside the ROBERT platform reachable space, therefore the chosen values were a minimum value $\delta = 0.01$ and the maximum $\delta = 0.1$.

However, due to the relative low amount of success results on the minimum value $\delta = 0.01$, it was assumed that the testing was not promising. The margin on the run δ values on the run were tested on $\delta = 0.3, 0.6$ because the second value is the double of the first one and it aimed to check if the fail ratio could be more related to the δ on the run than to the δ on the path correction. For this purpose, for the margin on the run $\delta = 0.6$ additional tests were run, being 13 in total. Given that the success results increased in an amount of 2 but the failed tests amount stayed the same, the minimum value $\delta = 0.01$ has been considered too bad because the total was of 6 successful tests, 9 acceptable and 8 failed, meaning that there is no clear majority on the successful or acceptable result values. Then, the tests were directly moved onto the next value, without testing other values for the margin on the run $\delta = 0.125, 0.75$.

In the case of using a path correction of a margin $\delta = 0.1$ the tests could be done for three different values for the margin on the run $\delta = 0.125, 0.3, 0.75$, doing 10 tests per every value, except for $\delta = 0.3$ that 9 tests were made because the battery of ROBERT ran out during the last test.

The graphs of the recorded tests have the same format and criteria as the tests with the virtual tube to determine if a tests is either successful, acceptable or failed. In this case, the values of rho_0 and $\delta\rho_0$ are not constant because not all the points are standing with the same height towards the virtual wall.

The following Figure 6.4 illustrates this with the graphs of three tests done with $\delta = 0.1$ on the path correction and then applied either $\delta = 0.125, 0.3, 0.75$ on the run:

- Top graphs: one of the tests with $\delta = 0.3$, which is a continuous increasing push. In this case, rho is always between rho_0 and $\delta\rho_0$, and the ratio λ has a very low value close to 0. The result of this test is successful.
- Middle graphs: one of the tests with $\delta = 0.75$, which is an intense increasing push towards and against the virtual wall. In this case, rho goes towards $\delta\rho_0$ and it overcomes it, but the distance towards the virtual constraint is at least $200mm$. On the right side, and the ratio λ increases accordingly and reaches λ_{max}. In total, the result of this test is acceptable.
- Bottom graphs: one of the tests with $\delta = 0.125$, which is a test with sporadic pushes. In this case, rho goes back and forth from the goal position, but in one of the pushes, it overcomes the margin distance $\delta\rho_0$ multiple times and also reaching distance values under 10mm. Therefore, the result of this test is failed.

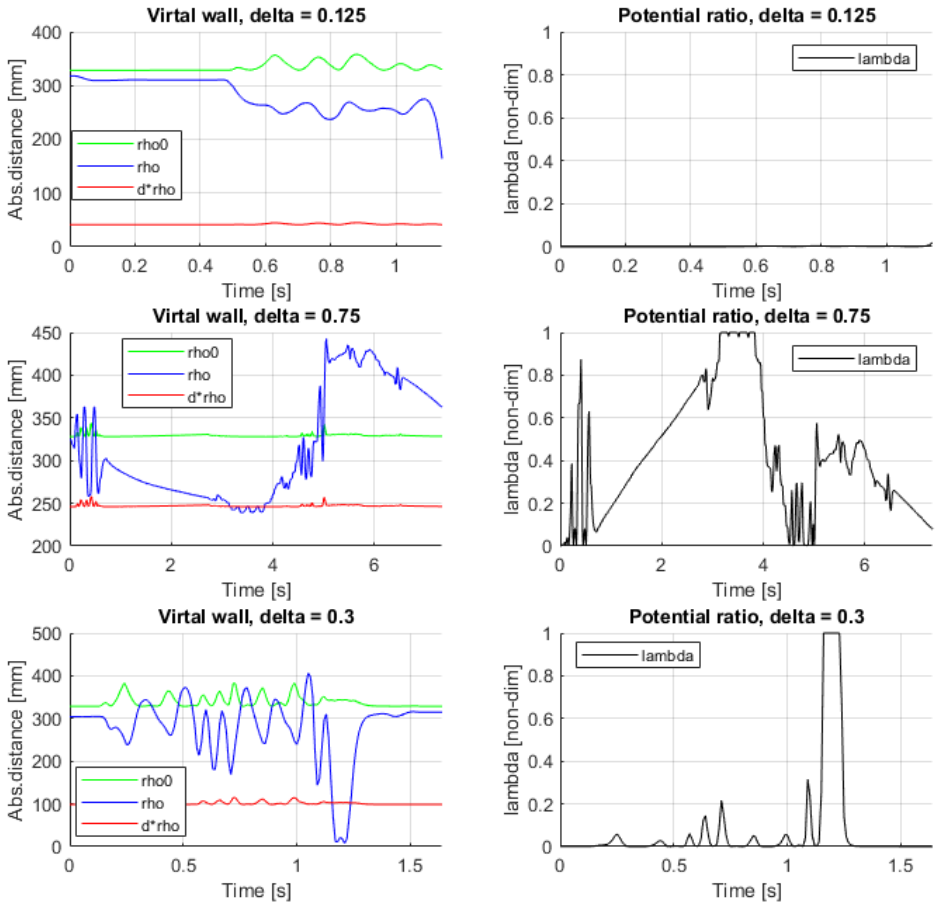

Fig. 6.4: Three test results for the potential ratio λ for a virtual wall with path correction $\delta = 0.1$. Left side: absolute distances and distance margins. Right: λ ratio. From top to bottom: test 5/10 for $\delta = 0.125$, test 7/9 for $\delta = 0.75$ and test 2/10 for $\delta = 0.3$.

Based on this, the results of all the tests are summarized as follows:

Table 6.5: Virtual wall test results for different δ for path correction and on the run.

δ(0,1) (path correction)	δ(0,1) (on the run)	Successful	Acceptable	Failed
0.01	0.3	2	4	4
	0.6	4	5	4
	Total =	*6 (26.07%)*	*9 (39.13%)*	*8 (34.82%)*
0.1	0.125	6	3	1
	0.3	2	5	2
	0.75	1	8	1
	Total =	*9 (31.03%)*	*16 (55.17%)*	*4 (13.79%)*

6.2.3 Velocity ratio

The other ratio mentioned on the APF is the velocity ratio β. The ratio represents how proportional is the velocity towards the maximum velocity allowed, in order to adapt the damping ratio on the impedance controller.

In this case the velocity ratio was tested the virtual wall and a path correction of $\delta = 0.1$, with 10 tests through the same trajectory for different velocity ratio margins δ_{beta}.

In order to determine a result from a test, the same criteria is applied as for the potential ratio λ. However, in this case the meaning of ρ_0 and ρ are different: as described in Equation 4.3.2, these parameters depend on the maximum and minimum velocities and the ratios are the difference between them. Thus, ρ is the velocity difference from the $\dot{x}_{max}$ to the current velocity of the end-effector, and $\delta\rho_0$ is the maximum velocity difference for setting the daming gain on the maximum value.

The following Figure 6.5 illustrates the interpretation better with two of the tests. And the result based on the graphs are described as follows:

- Top graphs: one of the tests with $\delta = 0.125$, which is a small push. In this case, rho is always between rho_0 and $\delta\rho_0$, usually close to rho_0, which means $rho_0 = \dot{x}_{max} - \dot{x}_{min}$, then $\rho \rightarrow \rho_0 = \dot{x} \rightarrow \dot{x}_{min}$. And the ratio β has a very low value close to 0. The result of this test is successful.

- Bottom graphs: one of the tests with $\delta = 0.75$, which is an intense short push and then a continuous push. In this case, rho goes towards $\delta\rho_0$ and it overcomes it, which means that during the push $\rho < \delta\rho_0 = \dot{x} > \delta\dot{x}_{max}$. On the right side, and the ratio β increases accordingly and reaches β_{max}. However, it is clear that ρ does not reach a lower value than $10mm/s$, therefore it does not violate the virtual constraint, which would be $\rho = 0$ implying that $\dot{x} = \dot{x}_{max}$. In total, the result of this test is acceptable.

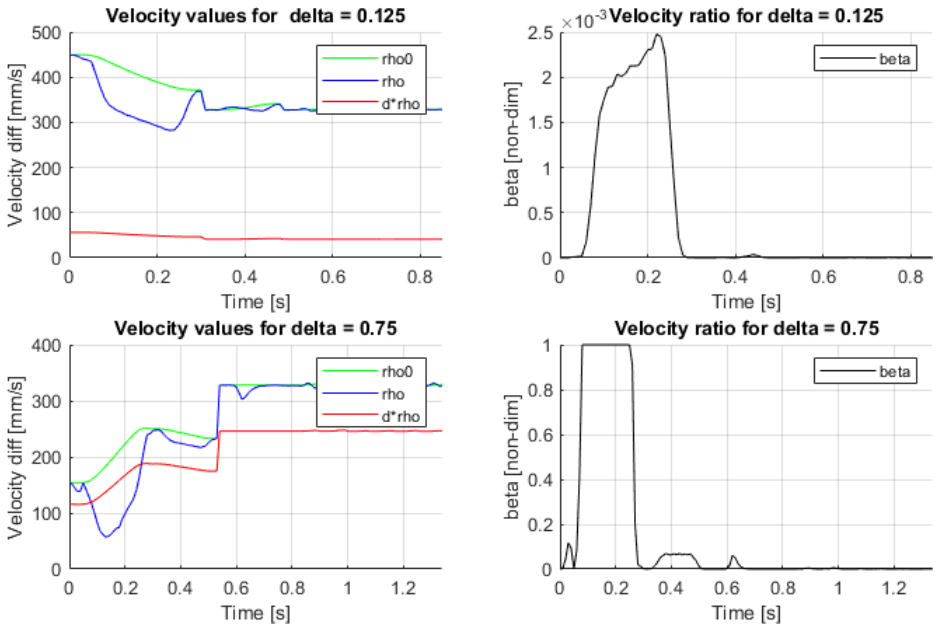

Fig. 6.5: Three test results for the velocity ratio β for a virtual wall with path correction $\delta = 0.1$. Left side: absolute difference velocity and velocity margins. Right: β ratio. From top to bottom: test 1/10 for $\delta = 0.125$, test 2/10 for $\delta = 0.75$.

Based on this concept, the following Table 6.6 summarize the results of all the tests for β:

Table 6.6: Virtual tube test results for different δ_β on the velocity ratio.

δ_β	**Successful**	**Acceptable**	**Failed**
0.125	8	2	0
0.3	0	10	0
0.75	1	9	0
Total =	9 (30%)	21 (70%)	0 (0%)

6.3 Complete approach

At the end of this project, all the presented methods and the implementation were tested as a whole. The aim was to test the complete application with the virtual wall and the virtual tube among with the force-velocity approach.

However, during the testing, the robot controller received quite often safety errors from the power module, which would require to re-start the controller. The concrete cause of this power error is unknown. It has been assumed that changing the controller gains and the velocity at the same time too drastically could have been the source of the power error. Due to this idea, additional changes were introduced in the software, by setting a limitation on the increment of the impedance controller gains to $500N/m$ in the axis gains, $50N/m$ in the angle gains and 0.3 on the damping.

Afterwards, only 6 tests were recorded in total without further issues, 3 of them with a virtual tube with a radius of $r_{vc} = 250mm$ and $r_{vc} = 500mm$. All of the tests have in common that the user pushed softly the arm through the path.

The following Figure shows the results of ρ within and the maximum and limit values $\rho_0, \delta\rho$ for both the distance towards the virtual constraint and the maximum velocity with the virtual tube of $r_{vc} = 250mm$.

In the case of $r_{vc} = 500mm$, the graphs show the same results. However, the ratio of success/acceptable/failed tests has not been applied to this section due to the lack of more tests and possible scenarios to have more reliable results.

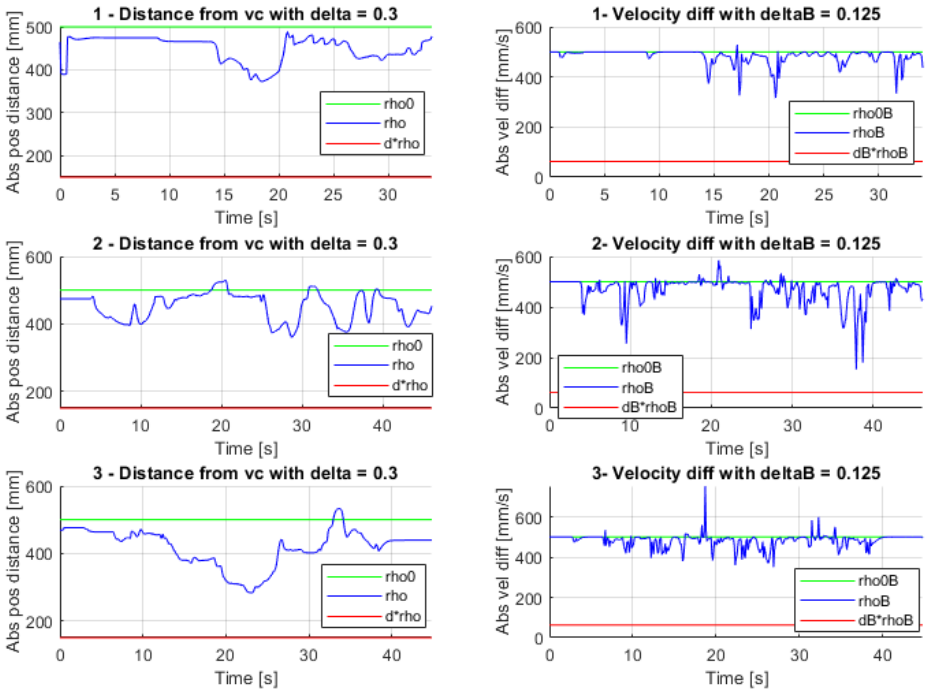

Fig. 6.6: Three test recordings for the end-effector positon and velocity within a virtual tube of $r_{vc} = 250mm$. Left graphs: distance to virtual constraint (blue), margin towards the constraint (red) and total distance from the center line to the virtual tube (green). Right graphs: velocity difference until reaching the maximum velocity (blue), margin towards reaching the maximum velocity (red) and minimum velocity (green). From top to bottom: test nr.1, 2 and 3.

7 Discussion

This chapter aims to look further into the proposed approach, the project implementation and the test results, including possible flaws on the system design and/or implementation.

On the proposed approach presented on Chapters 3 and 4, the APF is used for path correction and the corrected points are relocated depending on the ratio δ. This is a ratio that represents the proportional distance between the virtual constraint and the center x_{center}, which in this case happens to be an arbitrary point inside the allowed workspace. Depending on the chosen values for this reference point, the corrected path will be set accordingly. Therefore, in two separate scenarios with the same constraints, waypoints and δ, the corrected paths will be different. This fact might be uncommon to the reader, but it not an issue on the practical matter.

Another ratios have been introduced apart from δ, such as λ and β. In case of λ, it is a proportional and defined version of the proposed ρ by Khatib [9] This means that λ does not tend to infinite and that with a value of δ it is possible to determine when the maximum value is reached, which is very useful for the impedance controller gains, which are limited within a know range.

In a similar manner, β comes from the same concept, but used on velocity instead of position.

After the solution and methods were presented, the next step was the implementation, described in Chapter 5. Some incoveniences during the process:

- While implementing the hand-guiding recording and the trajectory updating functionalities, the recorded path was recorded in Joint space nad Cartesian space in separate files. Then, the robot had no issues with replicating a trajectory from a JS recordings worked perfectly, but in case of loading a CS path, the robot would not performed as expected, but the end-effector position would not be correct, but rather with a misplacement. This could be due to a mismatch on the reference frames while either saving or replicating the CS path, but on the code review the references matched while doing both processes. But in the end, the recording and loading paths feature work with recording and loading paths on Cartesian space .

- The robot in the simulation environment could only have a controller in JS or in CS, but not in both. And based on the previous point that the recorded paths are in JS, two robots were needed on the simulation environment. This fact is only a minor inconvenience, rather than an issue.

Moving onto the test results from Chapter 6, different types of tests are meant for different types of features:

- Force-velocity tests: they were done with only the linkage as end-effector and no using the boot and user manually pushed the end-effector through the path points with one hand. The force could not be recorded doing real lower-limb exercises with the boot and the lower limb due to the small space for the testing.

  Therefore, the limit values f_{min} and f_{max} are meant for this specific configuration of the end-effector and for the force of that single user, which means the limit values might differ in real exercises, where the patient pushes the robot with the lower limb with the boot.

  On the other hand, the results are promising. As seen on Figure 6.2, the velocity has a behavior similar to a "on-off" switch (either is close to 0 or it increases fast to 1): the robot shall move when the patient's push is detected and stay still when only the static force and sporadic peaks. Therefore, the robot reached the desired behavior on this feature.

- Artificial Potential Field(s) (APF) and virtual constraints: in both cases the successful and acceptable results sum the majority of outcomes (summing both outcomes are the $80\%, 83.33\%, 86.67\%, 86.2\%$ of the results for all the virtual tube tests and for the virtual wall with a path correction margin of $\delta = 0.1$).

  In the case of the virtual tube, concluding what the best settings are for the margin and radius δ, r_{vc} based on the results might be different depending on the reader. In one hand, the margin $\delta = 0.125$ had the biggest amount of success, but also the biggest amount of failed results. This is due to the distance left between the limit distance $\delta\rho$ and the actual constraint, which is in the case of a small value, the robot has a bigger room for motion, but in case of reaching the limit distance, it can be very short depending on the tube radius, and in the case of $r_{vc} = 125mm$, the margin distance $\delta\rho = 15.625mm$, and considering that the virtual constraint is violated under $10mm$, this leads to the failed results.

  On the other hand, the margin $\delta = 0.75$ has the biggest amount of acceptable results, but also the lowest scores on successful and failed results. This is because the limit distance $\delta\rho$ is the 75% of the total radius length, being this value the most restrictive out of al the tested δ values. But this setting also means that the λ ratio reaches its maximum when the robot has been misplaced by a 25% of the radius distance and therefore, keep pushing the robot towards the virtual constraint is more difficult, leading to lower values of failed tests.

  Overall, depending on the reading might be better the value of $\delta = 0.125$ or $\delta = 0.75$. In case of the radius r_{vc}, the values of $250, 500mm$ have very similar results for both cases of δ, being technically a radius of $500mm$ sightly better by having no failed results, while the radius of $250mm$ has one failed test. However, depending on the application and context, a radius of $500mm$ might not be practical.

  In the matter of tests on the virtual wall, the value of δ for path correction has been fundamental for the final results on the tests. But the location of the virtual wall was

not optimal and maybe by setting it a little bit lower it could have been possible to test other values for δ.

Overall, both virtual constraints have similar results and behavior under the same type of push from the user. Most of successful and acceptable results came from continuous and progressive pushes (as seen on the top and middle graphs on Figures 6.3 and 6.4), and most of failed tests happened while doing sporadic pushes (bottom graphs on the same figures). This could be due to the gains update time on the impedance controller. As mentioned in Chapter 5, the gains cannot be directly updated, but the motion must be cancelled and restarted. In the meantime, the robot is on a idle state and it is very easy to drag from the position it is standing. In the case that the user is doing a progressive push, the gains update progressively and if the robot updates the gains while being close to the virtual constraint, the current set up is already close to the maximum values and therefore, not easy to keep pushing. However, if the previous gain values were close to the minimums, then the robot is quite soft and therefore, very easy to make it go through a virtual constraint if the push is fast enough. This can be seen also on the Figures 6.3 and 6.4, where the robot was previously close to the goal position (rho_0 green line) and then it directly goes towards the virtual constraint, reaching distances under the margin.

Another inconvenience of fast-pushing the robot is that when the gains are updated drastically, the robot makes small violent drag when the adjustment is done. In case of running the controller in T1 mode, the safety would interrupt the program due to violating the maximum allowed Cartesian speed.

- Force-velocity and APF tests: as previously reported, power supply issues limited the testing of the complete system as a whole. The robot controller console showed the error as the power supply category, but it was catalogued as ”unknown error” and therefore, there is no concrete proof of where the error came from.

  It has been assumed that the complete system approach could be related because the power failures would happen only when an intense push towards the constraint would be applied on the robot, which increases exponentially both the velocity and controller motion settings at the same time, and the robot makes a harder drag movement when adjusting, which could lead to a physical current peak on the joint motors.

  After setting a limited increment on the gains values, the error would stop happening, but then the gains would not update fast enough when the robot was pushed towards the virtual constraints. Also the increments values were set in order to just keep the robot moving, but the values were not analysed enough in order to make them optimal or to verify if this possible current peak is the real cause of the power failure.

8 Conclusion

This chapter summarizes the whole project and the Research and Development Objective (RDO) or project goals.

Overall, this project is focused on the concept development for a method to retain the robot within an allowed workspace that is adaptive overtime in a changing environment.

On the first Chapters 2 and 3 the problem and solution strategies were analysed. The solution can be divided into three main parts: trajectory generation, virtual constraints and using a Artificial Potential Field(s) (APF) approach for retaining the robot within the allowed workspace.

The first part was focused on the trajectory generation, which would take into account the force feedback of the end-effector in order to generate the trajectory velocity accordingly. The goal of this part is to provide a trajectory that is based on the interaction between the patient and the robot, which is related to the goal RDO1.

The second part was focused on the virtual constraint settings. The concept of virtual constraints is not tied to only physical obstacles, but it can also represent and limit any other type of motion in either Joint space or Cartesian space. In this case, there are two types of virtual constraints: virtual walls that are a simplification of real physical objects and a virtual tube, which is wrapped around the the path and limits how much the patient can deviate the robot while doing an exercise.

The goal of this part is to provide an allowed workspace for avoiding crashes with objects and a space that helps the patient through the rehabilitation motions, which is related to the goal RDO2.

Last but not least, the third part of the project had the deepest research and most notable application: use of APF to maintain the robot within the restricted workspace. The presented approach has the property of being proportional and therefore, *elastic*, which means that the APF can be used for any workspace independently of its size or complex shape.

The goal of this part is to provide a method that allows to move the robot through the given workspace and generated trajectory from the previous two parts, so this area is related to the goal RDO3.

However, unexpected events happened on the testing and it was found that the implementation of this method has been completed with issues, such as the robot idle state while updating the gains or the power module failure due to an unkown reason.

Still the requirements were reached and the presented results prove it. Additionally, the results also look promising towards further development and improvement.

9 Future works

The presented work and research are not perfect and they can still be taken further and deeper. The following list are suggestions or ideas for future concepts:

- Improve the gains update overtime to avoid the power module error. This could be done either in high-level software, by programming a more complex algorithm that avoids the violent drag of the robot or in low-level software, by designing another controller that can already update the gains on the run without cancelling the motion.
- Design an integrated system within ROBERT to detect the obstacles and set the virtual constraints on the real world. This could be done either with a camera, LiDars or ultrasonic sensors on specific zones. This could reduce the complexity of the system by not having the APF duplicated in both simulation and reality.
- Keep improving the simulation. The amount of resources is limited and no projects are done at the moment with this robot using simulation. In the case of the simulator of CoppeliaSim, it has been proved to be quite complete and even if the simulation required two manipulators for one robot, it has been proved to be functional. Additionally, investing time in doing a better simulation could accelerate and be helpful for the developement of the previous two points.

References

[1] Henningk Agerm, Lukas Annwolf-Dieter, and Wahlster Wolfgang. "Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. industriellen Revolution". In: *VDI Nachrichten* (2016).

[2] Coppelia Robotics. *CoppeliaSim user manual*. https://www.coppeliarobotics.com/helpFiles/index.html. Accessed: 12-05-2021, 13:30.

[3] J.J. Craig. *Introduction to Robotics: Mechanics and Control*. John Wiley & Sons, 2013, p. 1. ISBN: 9781292052526.

[4] Iñaki Díaz, Juan Gil, and Emilio Sánchez. "Lower-Limb Robotic Rehabilitation: Literature Review and Challenges". In: *Journal of Robotics* 2011 (2011), p. 11. DOI: 10.1155/2011/759764.

[5] Carolina Gomez Salvatierra. *Project repository*. https://github.com/CG33569/LBR_interface. Accessed: 01-06-2021, 07:45. 2018.

[6] Jurgen Guldner and Vadim Utlun. "Sliding Mode Control for Gradient Tracking and Robot Navigation Using Artificial Potential Fields". In: *IEEE Transactions on robotics and automation* 11 (1995).

[7] Sebastian Hjorth et al. "An Energy-based Approach for the Integration of Collaborative Redundant Robots in Restricted Work Environments". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2019).

[8] Neville Hogan. "Impedance control: an approach to manipulation". In: *American Control Conference* (1984).

[9] O. Khatib. "Real-time obstacle avoidance for manipulators and mobile robots". In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. Vol. 2. 1985, pp. 500–505. DOI: 10.1109/ROBOT.1985.1087247.

[10] Katharina Kufieta. *Force Estimation in Robotic Manipulators: Modeling, Simulation and Experiments*. Tech. rep. Norwegian University of Science and Technology, 2014. URL: http://folk.ntnu.no/tomgra/Diplomer/Kufieta.pdf.

[11] KUKA. *KUKA Sunrise Cabinet Med - Instructions for Use*. KUKA Roboter GmbH, 2018, p. 107.

[12] KUKA. *KUKA Sunrise.OS 1.11 - Operating and Programming Instructions for System Integrators*. KUKA Roboter GmbH, 2016, p. 557.

[13] KUKA. *LBR Med - Instructions for Use*. KUKA Roboter GmbH, 2017, p. 129.

[14] KUKA. *LBR Med 14 R820 CAD model*. https://www.kuka.com/event/media?itemId=38D6BFA8F2F442F2B63B5BBD5060A8F2. Accessed: 26-04-2020, 21:30.

[15] Life Science Robotics. *Life Science Robotics: Meet ROBERT*. https://www.lifescience-robotics.com/meet-robert/. 2020.

[16] G. P. Moustris et al. "Evolution of autonomous and semi-autonomous robotic surgical systems: a review of the literature". In: *The International Journal of Medical Robotics and Computer Assisted Surgery* 7.4 (2011), pp. 375–392. DOI: https://doi.org/10.1002/rcs.408. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rcs.408.

[17] Sabudin elia nadira, Rosli Omar, and Che Ku Nor Hailma. "Potential field methods and their inherent approaches for path planning". In: 11 (Jan. 2016), pp. 10801–10805.

[18] B. Preising, T.C. Hsia, and B. Mittelstadt. "A literature review: robots in medicine". In: *IEEE Engineering in Medicine and Biology Magazine* 10.2 (1991), pp. 13–22. DOI: 10.1109/51.82001.

[19] S.Ho, R.Hibberd, and B.Davies. "Active compliance in robotic surgery: the use of force control as a dynamic constraint". In: *Proceedings of the Institution of Mechanical engineers* 211 (1997).

[20] S.Ho, R.Hibberd, and B.Davies. "Robot assisted knee surgery". In: *IEEE Engineering in medicine and biology* (1995).

[21] Mark Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Dynamics and Control*. John Wiley & Sons, 2004, pp. 193–219. ISBN: 9781119523994.

[22] Noriyuki Tejima. "Rehabilitation robotics: a review". In: *Advanced Robotics* 14.7 (2001), pp. 551–564. DOI: 10.1163/156855301742003. eprint: https://doi.org/10.1163/156855301742003.

[23] Prahlad Vadakkepat, Kay Chen Tan, and Wang Ming-Liang. "Evolutionary Artificial Potential Fields and Their Application in Real Time Robot Path Planning". In: *IEEE International Conference on Robotics and Automation* (2000).

[24] Federico Vicentini. "Collaborative Robotics: A Survey". In: *Journal of Mechanical Design* 143.4 (Oct. 2020). 040802. ISSN: 1050-0472. DOI: 10.1115/1.4046238. eprint: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/143/4/040802/6661214/md_143_4_040802.pdf. URL: https://doi.org/10.1115/1.4046238.

[25] Charles W. Warren. "Global path planning using artificial potential fields". In: *IEEE International Conference on Robotics and Automation* (1989).

[26] Wikipedia. *X-ray of a normal knee by lateral projection*. https://en.wikipedia.org/wiki/File:X-ray_of_a_normal_knee_by_lateral_projection.jpg. Accessed: 03-05-2021, 05:30. 2018.

[27] Han-ye Zhang, Wei-ming Lin, and Ai-xia Chen. “Path Planning for the Mobile Robot: A Review”. In: *Symmetry* 10.10 (2018). DOI: 10.3390/sym10100450. URL: https://www.mdpi.com/2073-8994/10/10/450.