Universidad Zaragoza

Trabajo Fin de Máster

DESIGN OF ROBOTIC SYSTEMS FOR HUMAN-ASSISTIVE TASKS

Author

Alessandro Giacalone Amorelli

Tutors

Gonzalo Lopez Nicolas

Ignacio Cuiral Zueco

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2024

Abstract

This thesis explores the design of robotic assistance tasks for humans using a simulation environment in MATLAB. The research focuses on two main areas: passive user scenarios utilizing potential field-based motion planning, and active user scenarios employing state space control methods. For passive user scenarios, a potential field algorithm is developed for safe and user-friendly collision avoidance in assistive tasks. This algorithm is applied to two key applications: robotic assistive feeding and dressing. The feeding task demonstrates the algorithm's effectiveness in navigating around static and dynamic obstacles, while maintaining a safe and efficient trajectory to the user's mouth. The dressing task adapts the algorithm to guide a simulated sleeve along a user's arm, taking into account potential arm movements. In active user scenarios, the research progresses to more complex state-space control strategies. A collaborative box transport task is simulated using direct imitation control with a Kalman filter to handle potential motor impairments of the assisted user, modeled as noise in the user's hand movements. Additionally, a glass of water manipulation task is addressed using Model Predictive Control for real-time trajectory tracking, incorporating the dynamics of water sloshing in the glass. The thesis provides a comprehensive analysis of each application, including performance metrics such as task completion time, collision avoidance, and trajectory smoothness. The results demonstrate the effectiveness of the proposed algorithms in various assistive scenarios, laying a foundation for future development toward real-world applications. This research contributes to the field of assistive robotics by addressing key challenges in human-robot interaction, real-time adaptation, and safety in close proximity to users. The simulation-based approach allows for rapid iteration and evaluation of different scenarios, paving the way for future implementation on physical robotic systems.

Key-words: Assistive robotics; Motion planning; Potential fields; Collision avoidance; Human-robot interaction; Robotic feeding; Robotic dressing; Kalman filter; Model Predictive Control; Simulation environment.

Contents

Introduction

In recent years, the field of assistive robotics has gained significant attention due to several pressing factors in our society. The aging population and shortage of healthcare workers have created an urgent need for technological solutions to support independent living and care for the elderly and disabled [1]. Additionally, there is a growing emphasis on enhancing the independence and quality of life for individuals with disabilities through advanced technological aids [2, 3]. These assistive robotic technologies have the potential to reduce the burden on caregivers while empowering individuals to lead more autonomous and dignified lives [4]. As a mechanical engineer, I recognized an opportunity to contribute to this important field by developing robotic systems that can assist humans in everyday tasks.

The current state of research in assistive robotics spans a wide range of applications, from robotic feeding systems to dressing assistance devices. Notable works in this area include the development of robotic arms for assistive feeding [5, 6] and the exploration of soft robotics for safe human-robot interaction in dressing tasks [7]. However, many existing solutions face challenges in terms of real-time adaptation to human movements, safety in close proximity to users, and intuitive control interfaces. These challenges have motivated my research to address these issues.

The main goal of this thesis is to implement and evaluate robotic assistance tasks for humans in a simulation environment using MATLAB. I focused on two key objectives throughout my research. First, I aimed to develop a motion planning algorithm based on potential fields for safe and user-friendly collision avoidance in assistive tasks with a manipulator. This algorithm forms the foundation for two subsequent applications explored in this thesis that are representative of passive user control scenarios.

In this part of the thesis, I implemented and analyzed a robotic feeding task, considering various scenarios such as static and dynamic obstacles, particularly focusing on the user's hands as potential impediments. This task required careful consideration of safety, efficiency, and user comfort in a complex and dynamic environment. Then, I designed and evaluated a robotic dressing assistance task. This application presented unique challenges in terms of trajectory planning and adaptation to the user's body

shape and movements.

In the second part of the thesis, I explored active user control scenarios, applying state-space control methods, and including tasks of collaborative box transport and glass manipulation with sloshing control. These scenarios addressed the challenges of simultaneous direct imitation of the user reference trajectory without considering system dynamics in one case and real-time Single Input - Multi Output MPC trajectory tracking while considering dynamics of the object glass filled with water in the other case. The box transport task, in particular, focused on dealing with potential motor impairments of the user, modeled as noise in the hand movements.

Throughout this thesis, I employed a simulation-based approach to test and refine the proposed algorithms. This approach allowed for rapid iteration and evaluation of different scenarios without the need for physical hardware, laying the groundwork for future implementation on physical robotic systems.

By addressing these objectives, my work aims to contribute to the advancement of assistive robotics and improve the quality of life for individuals requiring assistance in daily activities. In the following sections, we will go into the details of each objective, presenting the methodologies employed, the results obtained, and the insights gained from this research. The thesis will conclude with a discussion of the limitations of the current work and potential directions for future research in this rapidly evolving field of assistive robotics.

1. Passive user: Potential-field robot control with user-friendly collision avoidance

1.1 Potential fields and description of the controller

In robotic motion planning, particularly for manipulators, Potential Fields Theory offers an effective and computationally efficient method for achieving obstacle avoidance. The core idea is to generate an artificial potential field within the manipulator's workspace, where obstacles create repulsive fields and the goal position generates an attractive field. The end-effector of the manipulator moves in response to the gradient of these potential fields [8,9]:

$$F_{\text{att}}(q) = -\nabla U_{\text{att}}(q) \tag{1.1}$$

$$F_{\text{rep}}(q) = \nabla U_{\text{rep}}(q) \tag{1.2}$$

$$F_{\text{total}} = F_{\text{att}} + F_{\text{rep}} \tag{1.3}$$

- F_{att} and F_{rep} are the attractive and repulsive forces.
- U_{att} and ∇U_{rep} are the attractive and repulsive potentials.
- q is the current position of the manipulator's end-effector.

In this work, I designed a controller using Potential Fields within a Matlab simulation environment, modeling the potential field functions in velocity units [m/s]. The control action is determined by the sum of the various velocities generated by the potential fields acting on the system. The attractive and repulsive fields can take on different forms, and I will design their profiles to suit the specific application. I will apply this control logic to a manipulator, focusing on its end-effector, which will be subjected to a resultant 3D vector field, guiding it to the desired position.

1.1.1 Attractive field towards the attractive point

The attractive field's profile towards the endpoint of the trajectory is represented by the function γ, which defines the attractive velocity as a

function of the parametric scalar s. This scalar represents the normalized distance from the goal position and is directed along the vector v, i.e., towards the endpoint of the trajectory. The scalar s is defined as the ratio of the end-effector's distance from the starting point of the trajectory to the total distance between the starting point and the endpoint (the attractive point). The expression for s is given in equation 1.1.1, and its values are constrained within the interval $[0, 1]$.

Calling A the starting point, B the endpoint and X the end-effector's position:

$$s = \frac{\|A - B\| - \|X - B\|}{\|A - B\|} \tag{1.4}$$

For applications of human robotic assistance, a good choice for the shape of the potential field $\gamma(s)$ with respect to s is a smoothed trapezoidal, where the speed increases gradually from the starting point and then decreases gradually when approaching the endpoint. A minimum γ_{min} and maximum speed γ_{max} will be specified, the first one for ensuring that the end-effector will move towards the endpoint, the second one for safety and comfort reasons.

$$\gamma(s) = \begin{cases} \gamma_{min} + (\gamma_{max} - \gamma_{min}) \cdot \frac{1}{2} \left(1 - \cos\left(\frac{\pi s}{s_\mathrm{a}}\right)\right) & \text{if } s \in [0, s_\mathrm{a}), \\ \gamma_{max} & \text{if } s \in (s_\mathrm{a}, s_\mathrm{a} + s_\mathrm{c}], \\ \gamma_{max} + (\gamma_{min} - \gamma_{max}) \cdot \frac{1}{2} \left(1 - \cos\left(\frac{\pi(s - s_\mathrm{a} - s_\mathrm{c})}{1 - s_\mathrm{a} - s_\mathrm{c}}\right)\right) & \text{if } s \in (s_\mathrm{a} + s_\mathrm{c}, 1]. \end{cases} \tag{1.5}$$

Here, s_a, s_c, and s_d represent the acceleration, constant velocity, and deceleration phases, respectively, with total $s_\mathrm{total} = s_\mathrm{a} + s_\mathrm{c} + s_\mathrm{d} = 1$.

To properly design the function γ, I will need to select appropriate values for the parameters γ_{min}, γ_{max}, s_a, s_c, and s_d, based on the specific application requirements.

1.1.2 Repulsive fields on the obstacles

The repulsive fields are designed as a repulsive velocity function based on the distance between the end-effector and the obstacles. For the specific applications I worked on, I modeled the repulsive fields as the derivative of the sigmoid function, defined as follows:

$$\beta(x) = \frac{4 \cdot max_\beta \cdot \exp(-\theta_\beta \cdot x)}{(1 + \exp(-\theta_\beta \cdot x))^2} \tag{1.6}$$

The parameters I can adjust in this function are θ_β and $\max_\beta$, with the former controlling the width and the latter determining the maximum value of the function.
The repulsive function is designed as a function of the distance of the point X from the obstacle's surface, and there will be as many functions β as there are obstacles.

1.1.3 Control loop

Being X_i the position of the end effector at the $i-th$ time step, a discrete feedback control loop is applied:

The control input u_i is defined as:

$$\mathbf{u}_i = \mathbf{v}_i \cdot \gamma(s) + \sum_j \mathbf{w}_{ij} \cdot \beta_j(\|X_i - C_j\|) \tag{1.7}$$

The position update X_{i+1} is given by:

$$X_{i+1} = X_i + \mathbf{u}_i \cdot \Delta t \tag{1.8}$$

where:

- $\mathbf{v}_i$ is the normalized vector from X_i to towards the end point
- $\mathbf{w}_{ij}$ is the normalized vector from the center of the obstacle (C_j) to the end effector at instant i
- $\mathbf{u}_i$ is the control action (it is a resultant speed vector)
- Δt is the discrete time step

This equation is essentially an Euler integration [10], where the position is updated by adding the product of velocity and time step to the previous position. The accuracy of Euler's method depends on the time step Δt. A smaller Δt increases accuracy but also raises the computational cost. For this reason, I selected a value of Δt that balances accuracy and computational efficiency, as I will discuss later.

1.2 Risk evaluation of the algorithm and issues encountered with proposed solution

1.2.1 Alignment between obstacle and attractive point

When the center of the obstacle is aligned with the attractive point (i.e., the center of the obstacle lies on the segment AB), the current position of the end-effector is subjected to two forces directed along two opposite and aligned vectors. What I observed is that point X will progressively reduce its velocity towards the attractive point and eventually will not be able to advance further, becoming stuck around a certain position.

I solved this issue by imposing a condition in the main loop: if, for a fixed minimum number of iterations k, the speed of point X at iteration i is lower than the minimum speed of the end-effector scaled by a scalar r, an additional velocity is applied to point X. This velocity is directed along a vector $\mathbf{p}$, which lies in the plane perpendicular to the segment formed by point X, the repulsive point, and the attractive point.

The reason for waiting k iterations before introducing this additional velocity component is to account for scenarios where the obstacle may be moving. In such cases, waiting a few iterations may allow the points to become unaligned without the need for introducing a random velocity component to free the end-effector.Another important reason is to avoid the highly unlikely but very undesirable scenario where the random escape direction coincides with the obstacle's direction of movement, making it impossible to evade.
If for $k = 10$ iterations

$$\|X_i - X_{i-1}\| < r \cdot \gamma_{min} \cdot \Delta t \tag{1.9}$$

where $r = 0.3$, then

$$\mathbf{u} = 0.5 \cdot \gamma_{max} \cdot \mathbf{p} \tag{1.10}$$

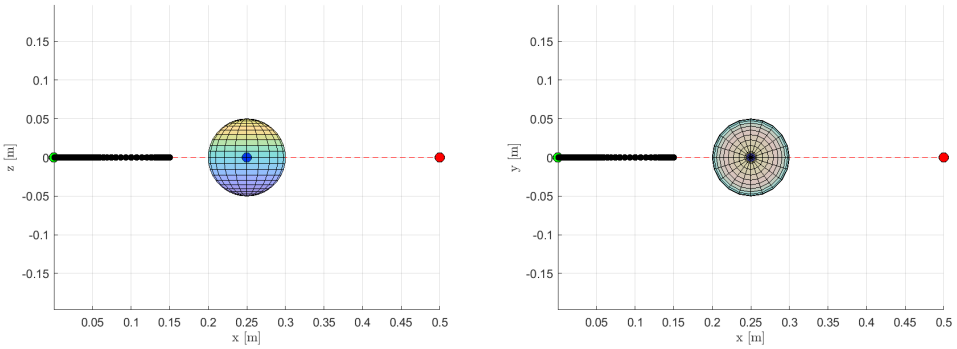


Figure 1.1: Obstacle avoidance algorithm without loop condition for alignment condition. The end-effector (its trajectory is defined by the black dots) remains stuck behind the the obstacle (sphere) and doesn't reach the endpoint (red point).

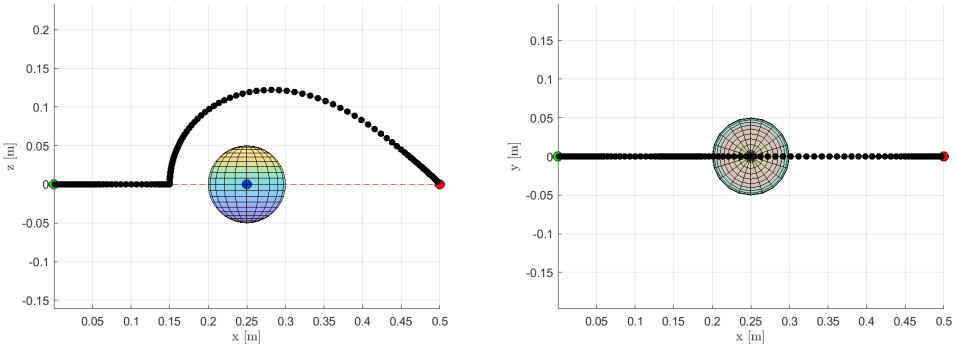


Figure 1.2: Obstacle avoidance algorithm with loop condition for alignment condition. The end-effector reaches the endpoint thanks to the loop condition.

1.2.2 Saturation of the control action

Another issue I encountered while designing the Potential Fields Algorithm is the saturation of the control action. Specifically, when point X is positioned between the obstacle and the attractive field, it is subjected to two forces in the same direction. This can be problematic if the obstacle is close to the attractive point, as the sum of the velocities from both the repulsive and attractive fields may cause point X to overshoot the attractive point. This is a critical issue in human assistance applications, as it could result in a collision with the human user. Ensuring safety for the user in such applications is imperative, so I thought of a method to address this problem.

The solution I developed is to apply a repulsive field, ϵ, on the user. For example, I simulated a scenario where the end-effector follows a linear trajectory aligned with an obstacle and the user's head, represented by two spheres, with the larger sphere representing the head. The obstacle and attractive point are placed very close together, which leads to the end-effector overshooting the endpoint and colliding with the user's head, as shown in Fig. 1.3. By introducing an appropriately constructed repulsive field on the surface of the user's head, the end-effector is prevented from colliding with the user, as seen in Fig. 1.4. In the figure, point X comes very close to the user's head after passing point B, but this is because I chose large values for the maximum and width of the function β to visualize the issue. With more appropriate parameter values for human assistance applications, this problem will occur rarely. Moreover, in this simulation, the attractive point is only five centimeters away from the user's head, whereas in our following applications, this distance will be greater.

Regarding the shape of the function ϵ, I did not use the derivative of the sigmoid as I did for the obstacles. Instead, I chose a profile that ensures the end-effector will never collide with the user, regardless of the value of the control action directed toward the user. This implies that the function tends to infinity on the user's head surface. A suitable function that meets these requirements is shown in equation 1.11, where x is the distance in meters from the surface of the user's head.

$$\epsilon(x) = \frac{1}{|W_\epsilon \cdot x^3|} \tag{1.11}$$

An important consideration in designing the repulsive field ϵ, and thus in selecting the parameter W_ϵ, is that its value at the attractive point must be less than the minimum speed of the function γ. Without obstacles, the end-effector's speed at the attractive point is equal to the minimum speed of γ. Therefore, to ensure that the end-effector reaches the endpoint, the value of the repulsive field ϵ at the attractive point must be lower than this minimum speed, as a higher value would prevent the end-effector from reaching its goal.

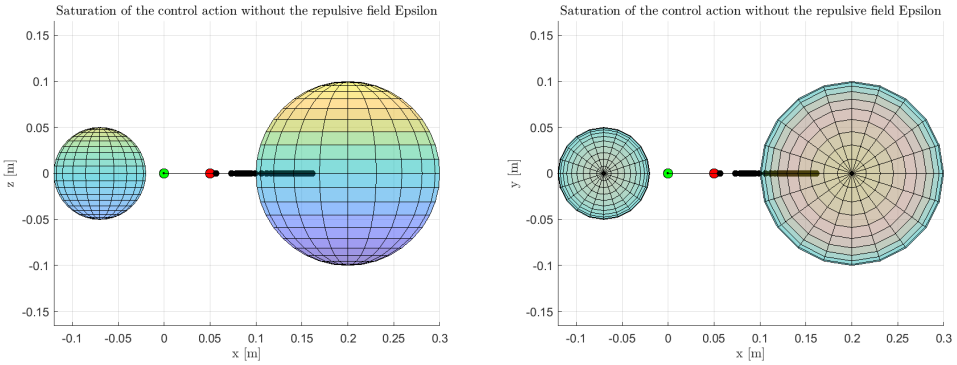


Figure 1.3: Saturation of the control action without repulsive field on the user's head. The smaller sphere on the left represents the obstacle, while the bigger one is the head of the user. The green point is the starting point A, the red one is the endpoint B and the the black points represent the points of the trajectory of the end-effector X. The end effector collides with the user.

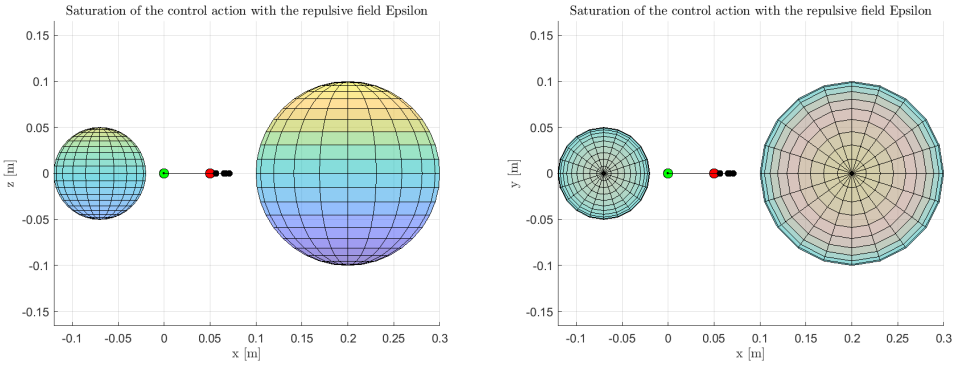


Figure 1.4: Saturation of the control action with a repulsive field ϵ on the user's head. The end-effector does not collide with the user.

1.2.3 Critical proximity of the obstacle to the endpoint and consequent task failure

Another critical scenario occurs when the obstacle is positioned very close to the endpoint B, within a critical distance that prevents the end-effector from reaching position B. This happens because, beyond a certain proximity to B, the repulsive field becomes stronger than the attractive field, causing point X to stop at a distance from B that exceeds the desired maximum range. Although this is not the ideal outcome, it remains safe for the user due to the function ϵ. We designed the system to prioritize collision avoidance over positional accuracy, so we accept this limitation. The reason for prioritizing obstacle avoidance is that a collision could compromise the integrity of the task, potentially causing the food to fall or creating a safety hazard for the user. While the end-effector's final position might be less comfortable for the user to reach, this scenario is expected to occur very rarely.

1.3 Example of application to human assistance task: Robotic Assistive Feeding

1.3.1 Definition of the Problem

The objective of my work on robotic assistive feeding was to ensure the success of robotic feeding in a simulated environment, with increasing levels of complexity. The simulation environment includes a model of a human with arms that are free to move, and a collaborative robotic manipulator, e.g., an UR10, instructed to bring food to the user's mouth. Fixed or moving obstacles, such as a bottle on a table, can be modeled and inserted into the simulation. However, to demonstrate the functionality of the motion planning algorithm I also implemented a dynamic scenario. I will focus on the user's hands as obstacles, as the arms and hands are the most likely to interfere during the feeding process. Furthermore, in this simulation, I assumed the torso and head remain fixed in space, disregarding any possible movement. I also did not consider the dynamics of the end-effector's gripping mechanism, limiting my study to the kinematics and motion planning of the manipulator, assuming it is already holding a piece of food cut to a size that can be easily swallowed by the user.

Human model

Only the upper body was modeled, as shown in Fig. 1.5, since it is the most relevant part of the body in a scenario where a person is sitting and being fed. The torso is represented by a cylinder, the head by a sphere, and the arms by two robotic 6-DOF kinematic chains (customized Puma 560 manipulators) with their base at the shoulder position. I sized the head radius, torso radius, and arm lengths based on average human body dimensions. As illustrated in the figure, the model is centered at the origin, with the arms aligned along the x-axis, and the positive y-axis representing the front of the human.

Measurement	**Value [m]**
Head Radius	0.1
Torso Length	0.5
Torso Radius	0.1
Upper Arm Length	0.3
Forearm Length	0.3
Hand Radius	0.075
Distance Torso-Shoulder	0.03

Table 1.1: Human Model Dimensions

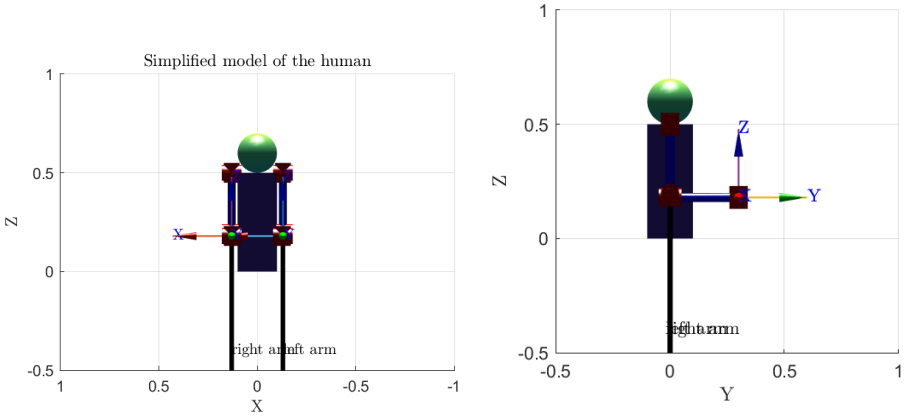


Figure 1.5: 3D representation of the model of the human that I used for the feeding tasks. It is modeled with a sphere, a cylinder and two kinematic chains. The axes are in meters.

Robotic manipulator

For the manipulator I also used a customized robot model as I did for the human arms, that I sized appropriately for the simulation environment. In Table 1.2 are shown the specifics of the manipulator.

Joints	Theta	d [m]	a [m]	Alpha [rad]	Offset [m]
1	q_1	0	0	$\frac{\pi}{2}$	0
2	q_2	0	0.3251	0	0
3	q_3	0.05	0.0203	$-\frac{\pi}{2}$	0
4	q_4	0.3251	0	$\frac{\pi}{2}$	0
5	q_5	0	0	$-\frac{\pi}{2}$	0
6	q_6	0	0	0	0

Table 1.2: D-H Parameters for the Robotic Manipulator

These specifics are adapted to the specific problem of feeding as I designed it.

Trajectory definition

The starting and final points of the end-effector trajectory will remain the same across all cases, with increasing levels of complexity. I will refer to the starting point of the trajectory as point A, where the end-effector is assumed to grab the piece of food to be fed, located on the table in front of the user. Point B represents the desired final position of the end-effector, placed 8 cm away from the user's mouth for both safety and comfort. I selected this

distance as a suitable trade-off between the user's comfort in reaching the food and ensuring the manipulator remains a safe distance from the user's face.

As shown in Fig. 1.6, point A is positioned at coordinates [0; 0.7; 0.3], and point B at [0; 0.18; 0.6]. These values are based on the fact that point B is at the same height as the center of the user's head and at a distance of 8 cm from the surface of the head. Point A is located on a table, 0.3 meters above the origin and approximately half a meter along the y-axis. I determined these values after experimenting with how a real application would be set up. The manipulator is also positioned on the table, midway along the trajectory, with a 30 cm offset in the x-direction from the plane containing points A and B.

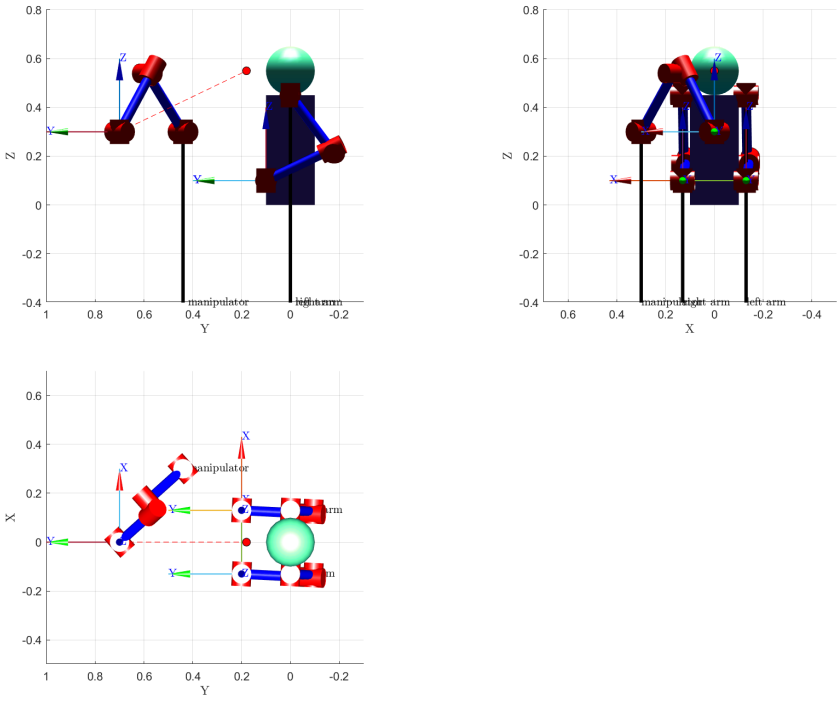


Figure 1.6: Environment of the Feeding problem with a human and a manipulator. The end-effector is placed at the starting point of the trajectory and the red point is the endpoint.

1.3.2 Potential fields considered in the feeding application

I designed the shape of the potential field functions such that the sum of all the potential fields ensures that the robot will never collide with the human, prioritizing safety. For the sake of representation, I will plot the profiles of the potential fields and the resultant field experienced by the

end-effector in the case with only one obstacle. Specifically, I will consider the configuration where points A, C, and B are aligned along a straight line passing through the center of the user's head. This configuration allows for an easily interpretable two-dimensional plot of the function σ, which represents the sum of all the previously described profiles.

With the center of the head positioned at [0; 0; 0.6], I selected point A at [0; 0.7; 0.6], point B at [0; 0.18; 0.6], and the center of the obstacle (the right hand) at [0; 0.45; 0.6] for the sake of visualization. These points define a straight horizontal line passing through the center of the user's head.

The parameter values for the potential field functions will remain the same as in the real configuration of the system, as described in the previous paragraph.

Attractive field at point B: Function γ

The function γ is directed along the vector $\mathbf{v}$, defined as:

$$\mathbf{v} = \frac{B - X}{\|B - X\|} \tag{1.12}$$

As described in Sec. 1.1, γ has the shape of a smoothed trapezoidal curve, defined between its minimum and maximum values, γ_{min} and γ_{max}. Its formula is given in equation 1.5. I selected a smoothed trapezoidal shape for the attractive potential to ensure user comfort and a smooth deceleration when approaching the user's mouth.

I chose a value of $\gamma_{max} = 0.1\,\mathrm{m/s}$, considering that in this particular scenario, the end-effector must travel a short distance for safety reasons and to avoid damaging the food or causing discomfort to the user. This value can be increased depending on the criticality of the application. Additionally, I selected a minimum speed, γ_{min}, of $0.02\,\mathrm{m/s}$ to ensure that the end-effector engages motion towards point B. The function γ is plot in Fig. 1.7.

For the acceleration, deceleration, and constant speed phases, I chose the parameters s_a and s_d equal to 0.25 to ensure a gradual variation of speed, and s_c equal to 0.5.

Parameters of the Gamma Function	**Value**
γ_{min} [m/s]	0.01
γ_{max} [m/s]	0.1
s_a	0.25
s_d	0.25
s_c	0.5

Table 1.3: Parameters of the γ function

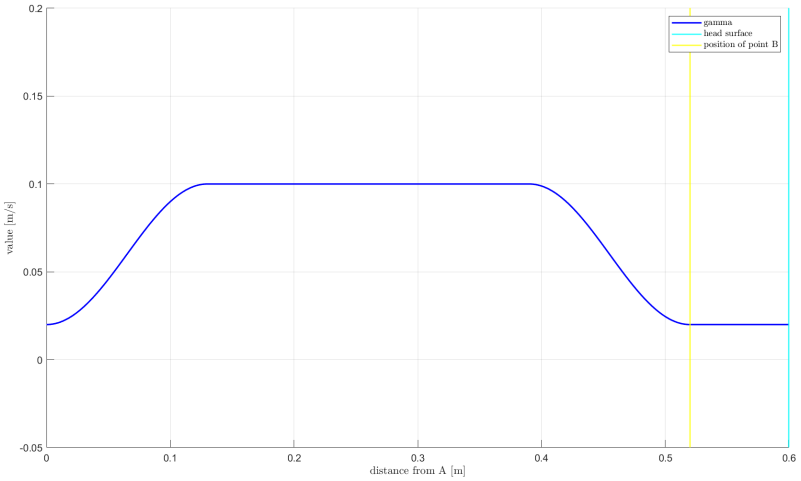


Figure 1.7: Function γ (blue curve), i.e. attractive field towards the target. The yellow and cyan vertical lines represent respectively the target B and the head's surface we have to avoid. For clarity, this is a 1D representation of the 3D problem.

Repulsive field on the obstacles: Function β

The function β is applied in the direction defined by the vector $\mathbf{w}$:

$$\mathbf{w} = \frac{X - C_r}{\|X - C_r\|} \tag{1.13}$$

where C_r is the position of the center of the right hand of the user. As discussed earlier, the function β is shaped like the derivative of a sigmoid, with max_β and θ_β as variable parameters. Its formula is given in equation 1.6.

I designed β by considering the hand as a sphere with a diameter of 15 cm, setting the maximum value of β on the surface of the sphere. At a distance of 3 cm from the hand's surface, β equals γ_{max} to keep the end-effector at least 3 cm away from the hand in a static scenario. Given $\gamma_{max} = 0.1\,\text{m/s}$, a reasonable value for max_β would be $0.2\,\text{m/s}$. The value of max_β is crucial, as it determines the maximum speed at which the user's hands can move toward the end-effector before a collision occurs. In dynamic cases, where obstacles are in motion, this ensures the end-effector can avoid obstacles moving toward it at speeds up to max_β. We will discuss this further when analyzing system performance. Thus, the values of parameters of the function β, represented in Fig. 1.8, in this context are:

$$max_\beta = 0.2 \tag{1.14}$$

$$\theta_\beta = 44 \tag{1.15}$$

We can create additional repulsive functions for the user's arm axis, torso, and the manipulator's axis to ensure collision avoidance. However, I simplified the system for clarity. I will apply a repulsive field along the entire arm axis for the dressing task.

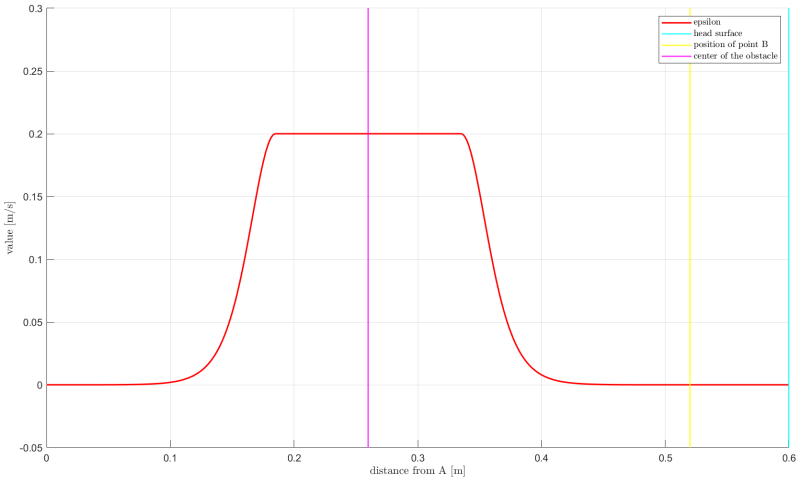


Figure 1.8: 1D representation of function β (in red), i.e. obstacle repulsive field. The pink, yellow and cyan vertical lines represent respectively the center of obstacle of 0.15 meters in diameter, the target B and the surface of the head of the user.

Repulsive field on the head surface: Function ϵ

The direction of the field ϵ is defined by the vector $\mathbf{q}$:

$$\mathbf{q} = \frac{X - C_{\text{head}}}{\|X - C_{\text{head}}\|} \tag{1.16}$$

where C_{head} is the center of the user's head.

The repulsive field on the head's surface is designed to avoid saturation of the control action, as explained in Sec. 1.1.2. The formula for ϵ is given in equation 1.11. Its value is infinite on the head's surface, and its profile is steep, as it is intended only for safety. The field tends to infinity on the head's surface to ensure that, regardless of the number of obstacles acting on the end-effector in the direction of the head, the end-effector will never collide with the user.

To illustrate this, suppose there is no attractive field γ, but only n obstacles with the same repulsive function β, each nearly touching the end-effector on the opposite side of the head. In this case, the control action

at instant k due to the obstacles would be equal to the sum of the maximum values of β, as shown in equation 1.3.2:

$$\mathbf{u}_k = \sum max_{\beta_i} \cdot w_i = n \cdot max_{\beta_i} \tag{1.17}$$

Thus, a profile for ϵ that tends to infinity ensures no collision between the end-effector and the head, regardless of the number of obstacles.

The design of the function parameters must ensure that at point B, ϵ is smaller than γ_{min}; otherwise, point X will never reach point B. The shape of ϵ guarantees that X will not collide with the user's head. As shown in equation 1.11, the parameter W_ϵ is crucial. I chose a value of $W_\epsilon = 200000$ to ensure a steep curve, with a value of approximately 0.01 m/s at a distance of 8 cm, where point B is located:

$$W_\epsilon = 200000 \tag{1.18}$$

$$\epsilon(0.08\,\mathrm{m}) = 0.01\,\mathrm{m/s} \tag{1.19}$$

As with the repulsive field for obstacles, we can also apply the safety repulsive field ϵ to the user's torso.

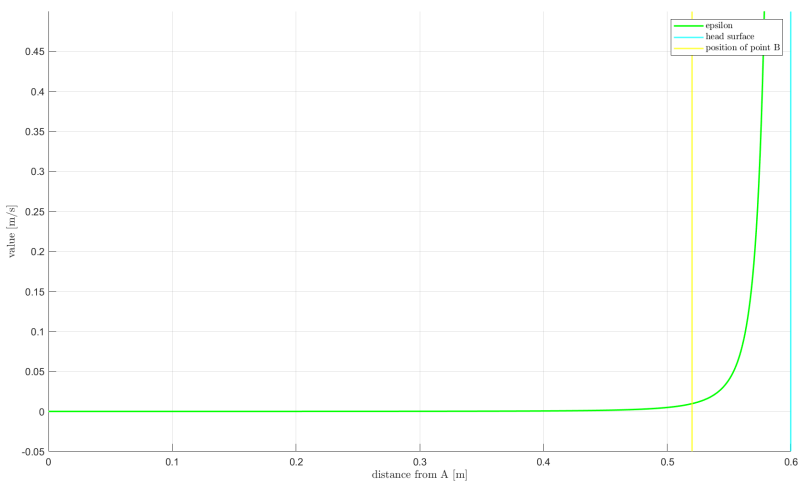


Figure 1.9: 1D representation of function ϵ, i.e. repulsive field on the user's head.

Resultant potential field on the end-effector: Function σ

The resultant potential field σ is directed as the sum of vectors $\mathbf{v}$, $\mathbf{w}$, and $\mathbf{q}$. In Fig. 1.10, we can see the resultant potential in terms of velocity experienced by the end-effector, as a function of the distance from point

A, in the case of a single static obstacle (e.g., the right hand) aligned with the specified points. The negative values represent the attractive potential, where the end-effector is drawn towards, while the positive values represent the repulsive potential from the obstacles.

$$\sigma = -\gamma + \beta + \epsilon \tag{1.20}$$

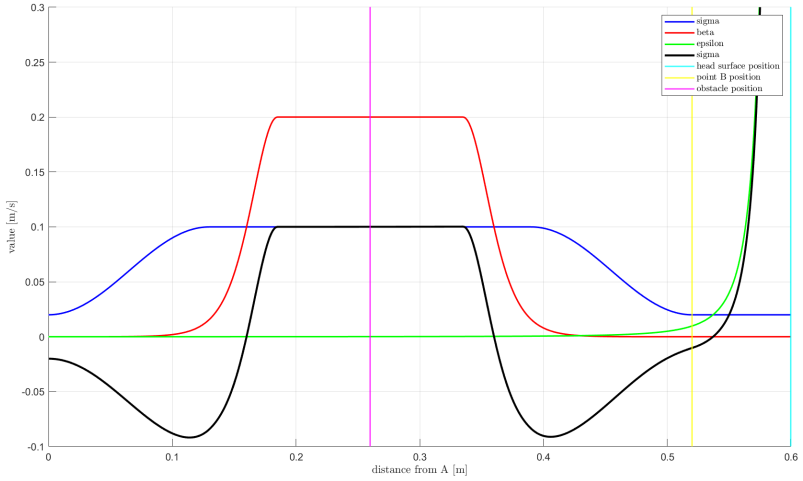


Figure 1.10: Resultant potential field, function σ(in black). This a simplified 1D representation of the 3D problem, just for visualization purposes. The resultant function σ is the sum of attractive and repulsive functions γ, β and ϵ. The resultant potential field in a certain position is attractive if σ is negative and repulsive if σ is positive.

1.3.3 Control loop

In our specific application with two obstacles and the safety repulsive field ϵ, our controller is described by the equations 1.21 and 1.22.
The control input u_k is defined as:

$$\begin{aligned}\mathbf{u}_k = &\mathbf{v}_k \cdot \gamma(s_k) + \mathbf{w}_{l(k)} \cdot \beta_l(\|X_k - Cl_k\| - R_{hand}) \\ &+ \mathbf{w}_{r(k)} \cdot \beta_r(\|X_k - Cr_k\| - R_{hand}) + \mathbf{q} \cdot \epsilon(\|X_k - C_{head}\| - R_{head})\end{aligned} \tag{1.21}$$

The position update X_{k+1} is given by:

$$X_{k+1} = X_k + \mathbf{u}_k \cdot \Delta t \tag{1.22}$$

where:

- β_l and β_r are the repulsive functions on the left and right hand respectively
- $\mathbf{w}_l$ and $\mathbf{w}_r$ are the directions of application of the repulsion on the end effector
- R_{hand} and R_{head} respectively are the radius of the hands and of the head, both modeled as spheres
- Cl_k and Cr_k are the positions of the centers of the left and right hand at the instant k
- C_{head} is the center of the user's head
- X_k is the position of the manipulator's end effector at the instant k
- $\mathbf{u}_k$ is the control action at the instant k

We will now apply the motion planning algorithm to feeding tasks with increasing levels of complexity, starting from simple feeding with no obstacles, progressing to feeding with one static obstacle, then feeding with two static obstacles, and finally feeding with one moving obstacle. These scenarios represent potential configurations in a real application and are designed to test the performance of the system.

To simplify the problem, for all feeding sub-cases, I chose the same starting and goal positions, A and B, while assuming the user's head and torso remain still. Only the user's hands are considered to move. We will analyze the performance and effectiveness of our solution across all cases, using the simplest case of feeding without obstacles as the baseline. For performance analysis, I will describe the parameters used in my approach.

1.3.4 Performance analysis

The performance parameters I used in my analysis are:

1. Sample time Δt
   The sample time Δt is a crucial parameter that affects the performance and effectiveness of our system. Specifically, it influences several key aspects:

   - **Smoothness of the trajectory**: A smaller Δt implies a smoother trajectory, as there will be smaller jumps between trajectory points.
   - **Completion time of the task**: A smaller Δt allows for a smoother trajectory and a more responsive system, leading to shorter completion times.
   - **Accuracy in reaching the endpoint** B: Smaller sample times result in higher accuracy.
   - **Computational cost**: Smaller sample times increase computational cost, which can be an issue for real-time applications like robotic assistive feeding.

   The sample time should balance system performance and computational cost. Robotic assistive feeding systems must operate in real-time, responding quickly to user movements and environmental changes. High computational costs can cause delays in processing sensor inputs and executing control actions, potentially resulting in slower response times and a less intuitive user experience. Therefore, selecting the correct value for the sample time is critical.

   In real applications, Δt should be chosen based on sensor and actuator capabilities, as well as environmental dynamics. In my simulation environment, I decided that 0.1 seconds is a reasonable value, as this produced good results, while smaller values caused excessive computational load on my computer. In real applications, smaller values in the order of 0.01 seconds would be more appropriate.

2. Accuracy
   Accuracy is defined as the distance between the final point of the end-effector's trajectory and the attractive point B. In the algorithm, the main loop condition is that the distance $\|B - X\|$ is less than or equal to a predefined threshold b. For this application, which requires high accuracy, b should be on the order of millimeters to ensure the user's safety and comfort in reaching point B with their mouth. The value of b depends on the sample time and the environment.

   For the chosen sample time of 0.1 seconds, a reasonable value for b is greater than $\gamma_{min} \cdot \Delta t$. We will set b to double that value, i.e., 0.004

meters (4 millimeters), to account for possible small contributions from the repulsive fields β. Therefore, in all obstacle configurations, except for the critical case where an obstacle is very close to the endpoint (as discussed in section 1.2.3), the algorithm ensures that $\|B - X\| \leq b$.

A smaller value of b would result in better precision but may also increase task completion time and lead to instability or excessive corrections as the end-effector approaches the target. In a practical context, the system may be affected by mechanical limitations, such as backlash in the joints or precision limits of the sensors and actuators, and should account for external factors such as computational delays or environmental disturbances that could impact the minimal achievable b.

3. Time efficiency
   This is measured as the time required to complete the task. We will compare this metric to the baseline case of feeding without obstacles.

4. Smoothness of the trajectory
   Trajectory smoothness is an important parameter, as it relates to the predictability of the robot's actions and the user's comfort. The smoother and more predictable the trajectory, the safer and more comfortable the user will feel.

1.3.5 Feeding, still person, no obstacles

In this baseline scenario, the end-effector has to move from point A to point B without having to deal with the interaction with obstacles. The user's arms are placed far from the zone of interest.
In Fig. 1.11 we can observe the trajectory of the end effector. The green and red points represent the points A and B, respectively, where as the black circles represent the points of the end effector's trajectory. As expected, the manipulator moves in a linear trajectory.

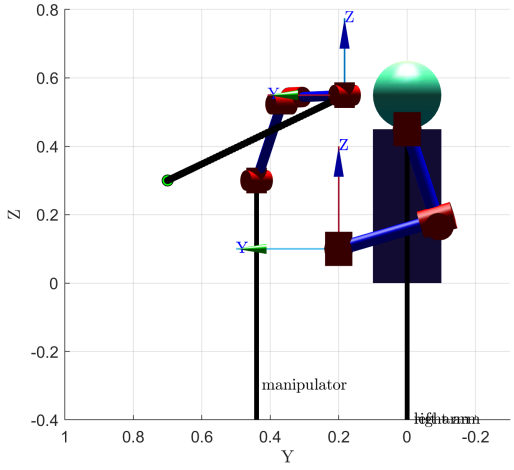


Figure 1.11: Static assistive feeding without obstacles. The figure shows an assistive robot, a human and the simple straight trajectory of the end-effector from the starting point (green point) to the endpoint (red point). The trajectory is represented by the black points, positions taken by the end-effector at every instant. The snapshot was taken at the end of the simulation.

In Table 1.4, the performance of the task is reported. The task is considered successful if the end-effector reaches point B within a range of 4 millimeters (the value of b). For this baseline case, where no obstacles affect the manipulator's trajectory, the system successfully feeds the user in 9.7 seconds, which is a very good result, although this is a simulation and the time required for grabbing the piece of food is not considered.

Naturally, without the presence of obstacles, the end-effector follows an undisturbed linear trajectory from point A to point B. As a result, the trajectory is very smooth and predictable, ensuring the user feels safe and comfortable throughout the feeding process.

Success	yes
Completion time [seconds]	9.7
Collision with obstacles	-

Table 1.4: Static Feeding effectiveness without obstacles

1.3.6 Feeding with one hand of the user in the way: Fixed hand position in space

Now, a single fixed obstacle is added, representing the user's right hand. In this sub-case, we will test the algorithm for different hand positions: the hand lying on the segment AB, the hand close to the starting point, the hand close to point B, and the hand at a critical distance from the endpoint. These scenarios represent cases where the user still wants the manipulator to feed them but has inadvertently placed their hand in the way. While these are rare situations, they are included to test the algorithm's robustness. The fourth scenario illustrates the risk discussed in section 1.2.3. We will refer to the position of the center of the user's right hand as C_r.

1. **Right hand close to the starting point A:** $C_r = [0, 0.56, 0.31]$
   The arm is almost fully extended to position the hand center as close as possible to point A, near the segment AB. As shown in Fig. 1.14, the obstacle immediately influences the trajectory, which remains relatively smooth and predictable. The system succeeds in feeding the user.

2. **Hand aligned with points A and B in the middle of the trajectory:** $C_r = [0, \frac{A_y+B_y}{2}, \frac{A_z+B_z}{2}] = [0, 0.440, 0.425]$
   This case tests the loop condition I designed for situations where the repulsive and attractive fields become equal and opposite after several iterations, preventing the manipulator from completing the task. The hand is placed exactly in the middle of segment AB, and Fig. 1.13 shows a rough trajectory with a sudden change in direction due to this setup. In contrast, Fig. 1.12, where the hand is placed roughly in the middle but not directly on the AB segment, shows a much smoother and more predictable trajectory.

   As shown in Fig. 1.13, the system successfully feeds the user within the desired range of point B, without colliding with the environment.

3. **Hand close to the endpoint B:** $C_r = [0, 0.29, 0.46]$
   The center of the obstacle is placed near the endpoint, just below the segment AB, causing the trajectory to deviate as the end-effector approaches the destination. This is likely the case where the user feels the least comfortable due to significant changes in direction close to the endpoint. However, once the end-effector overcomes the obstacle, the trajectory becomes relatively smooth and predictable. The sequence of snapshots of the simulation is shown in Fig. 1.15.

4. **Hand at a critical distance from B:** $C_r = [0, 0.26, 0.5]$
   In this scenario, the end-effector tends to settle in an equilibrium position where the potential fields cancel each other out, preventing the condition $\|B - X\| < b$ from being satisfied.

The simulation results are presented in Table 1.5. In the first three configurations, the system successfully feeds the user without colliding with the obstacle, although with higher completion times compared to the baseline case with no obstacles, which is expected. The best result is achieved when the obstacle is placed close to point B. However, when the obstacle is too close to B, as in the fourth configuration, the manipulator is unable to reach the final position with the desired accuracy.

	Close to A	On AB	Close to B	Crit. dist. from B
Success	yes	yes	yes	no
Completion time [s]	15.9	15.4	**13.7**	-
Collision	no	no	no	no
$\|\|B - X\|\|$ [mm]	$< b$	$< b$	$< b$	44.62

Table 1.5: Effectiveness of the system with one obstacle in different critical cases

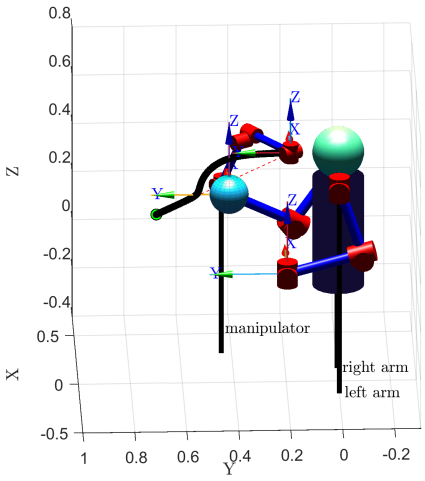
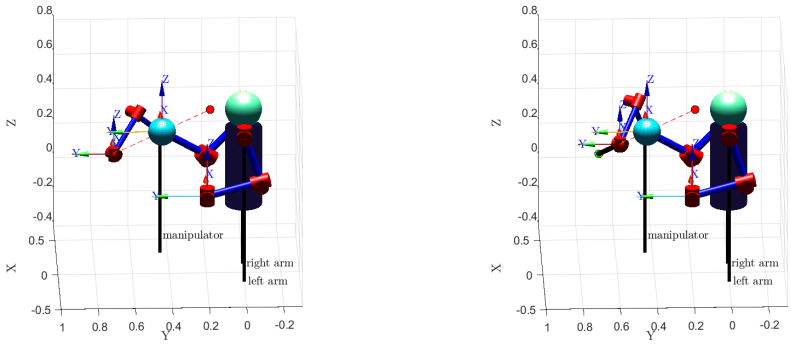
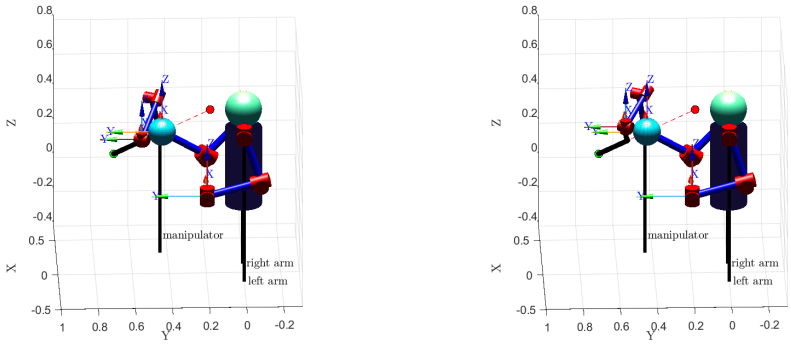


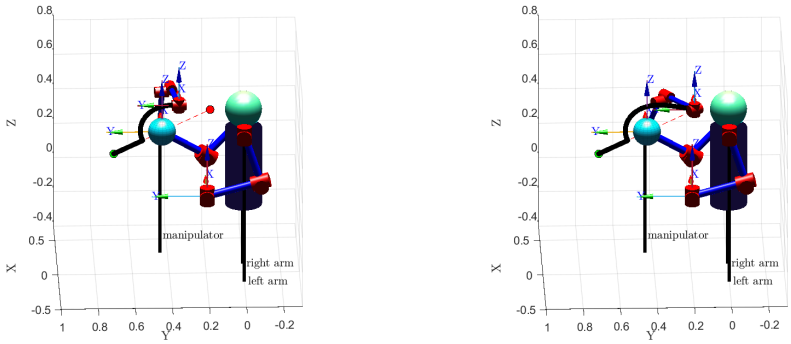
Figure 1.12: Assistive feeding with user's hand positioned in a non critical position, ensuring a smooth and predictable trajectory



(1) (2)



(3) (4)



(5) (6)

Figure 1.13: Static Assistive Feeding sequence of snapshots of simulation with the right hand on the segment AB. This particular condition initiates the Loop alignment condition 1.2.1.

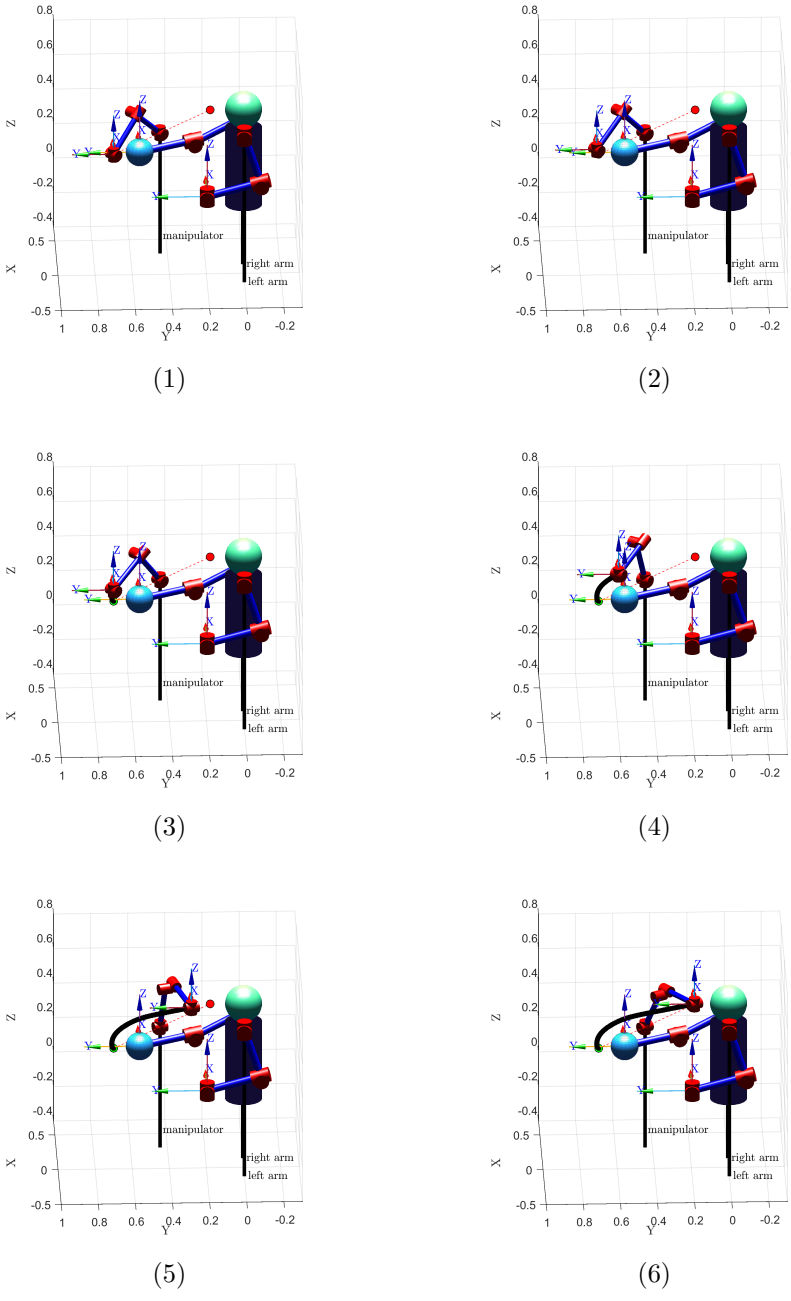
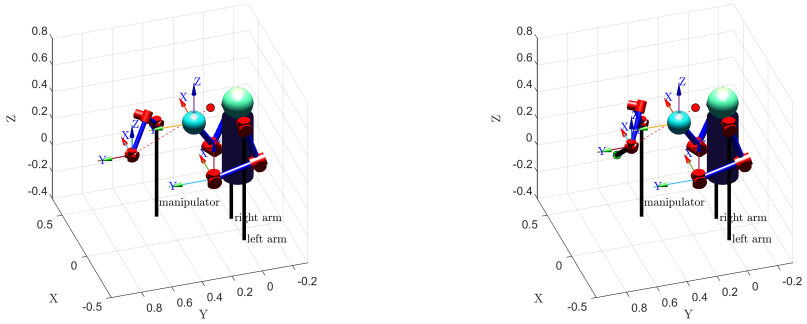
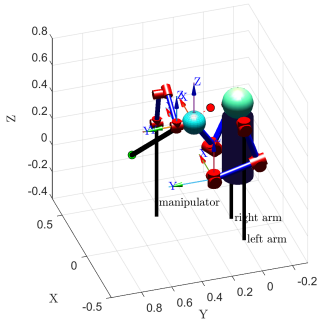


Figure 1.14: Static assistive feeding sequence of snapshots of simulation with the right hand close to the point A. The obstacle immediately interferes with the trajectory.

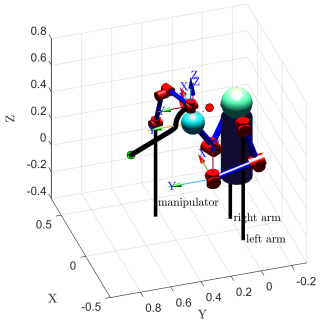


(1)

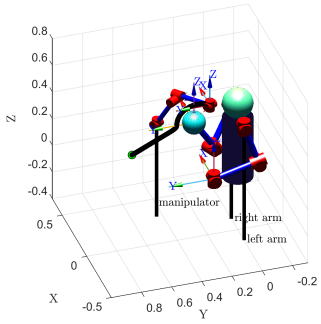
(2)



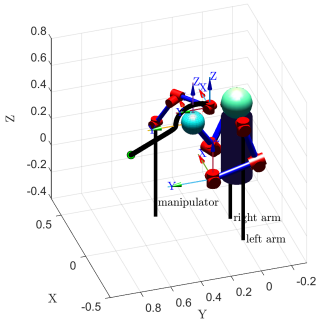
(3)



(4)



(5)



(6)

Figure 1.15: Static assistive feeding sequence of snapshots of simulation with the right hand close to the point B. The obstacle interferes with the trajectory in proximity of the endpoint.

1.3.7 Feeding with both hands of the user interfering with the end effector and fixed position of the hands in space

In this scenario, both of the user's hands are fixed in space and influence the trajectory of the end-effector. The system's goal is to feed the user while avoiding collisions with both obstacles, testing the algorithm in a more complex situation. We will simulate a single case where the two obstacles are positioned close to each other, near the segment AB, and on opposite sides of the segment.

- Position of the right hand: $C_r = [0.05, 0.5, 0.34]$
- Position of the left hand: $C_l = [-0.05, 0.35, 0.55]$

The trajectory in 3D is shown in Fig. 1.16 and the results of the simulation are:

- Successful completion of the task with the desired accuracy
- No collision with any of the obstacles
- Completion time of the task of 13.5 seconds.

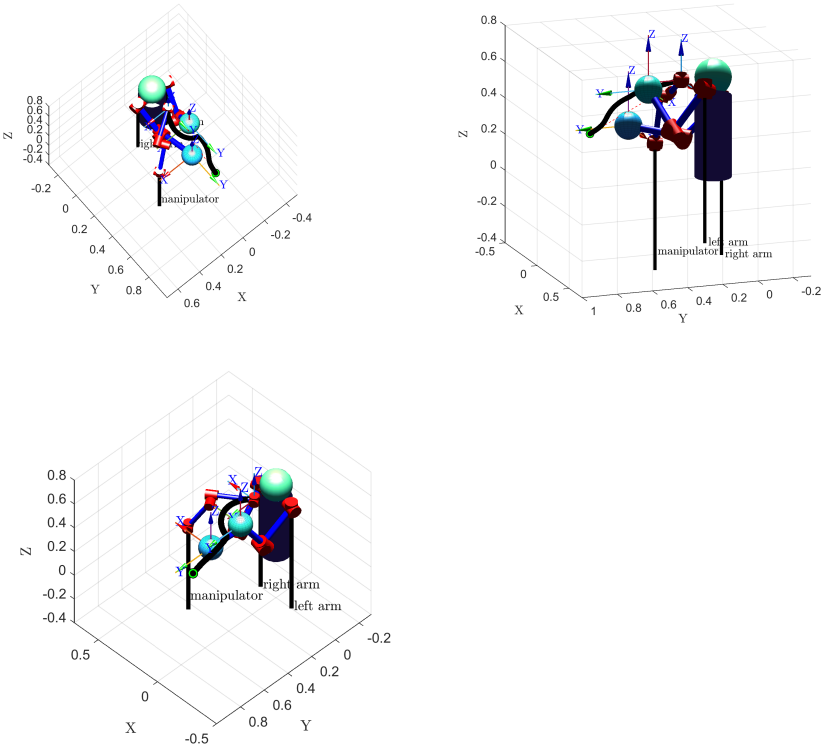


Figure 1.16: Assistive feeding with both user's hands affecting the trajectory

1.3.8 Feeding, user moving the right hand

In this scenario, I modeled the dynamic movement of the user's right hand with a cyclical motion in the middle of the segment AB. Although such motion is unlikely for this type of application, involuntary movements of the user's hand can complicate the manipulator's trajectory, and a cyclical motion around the segment AB presents a challenging test for our system. Specifically, the obstacle moves along the z-axis in a sinusoidal pattern. The motion of the right hand is defined by the following equations:

$$\begin{cases} \mathbf{C_r}(t) = C_{r0} + [0, 0, m_{amp} \cdot \sin(f \cdot t)] \\ \dot{\mathbf{C}}_\mathbf{r}(t) = [0, 0, m_{amp} \cdot f \cdot \cos(f \cdot t)] \end{cases} \tag{1.23}$$

where

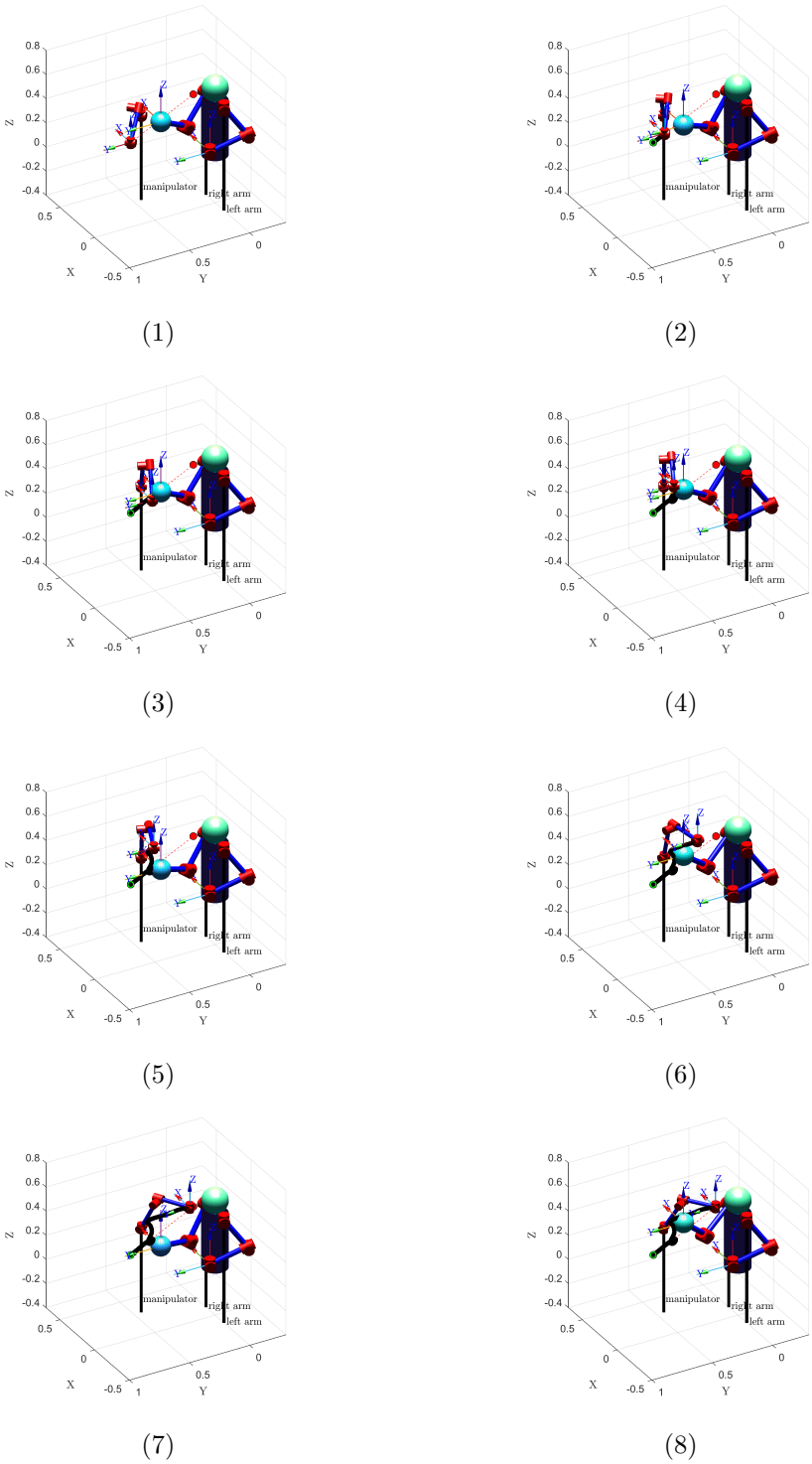
- $C_{r0} = [0, 0.45, 0.4]$
- $m_{amp} = 0.1$ m
- The value of f is directly linked to the speed of the obstacle. In fact, the maximum speed of the obstacle is $\|\dot{\mathbf{C}}_\mathbf{r}\|_{max} = m_{amp} \cdot f$.
  We will increase the value of f until a collision occurs with the obstacle. Since $max_\beta = 0.2\,\text{m/s}$, the theoretical maximum value of f for which the algorithm should succeed in most configurations is the one for which $m_{amp} \cdot f = 0.2\,\text{m/s}$.
  This gives $f = 2$.

The results of the simulation for different speeds of the obstacle are shown in Table 1.6. As predicted, increasing f increases the speed of the obstacle until a point where collision becomes inevitable. The best performance of the system is achieved at lower obstacle speeds.

	f=1	$f = 2$	$f = 3$	$f = 4$
Success	yes	yes	no	no
Completion time [seconds]	**10**	13.4	-	-
Collision with obstacles	no	no	yes	yes

Table 1.6: Effectiveness of the system with one oscillating obstacle for different values of f

The sequence of the animation with $f = 2$ is shown in Fig. 1.17. Due to the relatively fast and cyclical motion of the hand, the end-effector is forced to follow a complex trajectory that is not smooth and difficult to predict, yet it still successfully completes the task in 13.4 seconds.



(1) (2)

(3) (4)

(5) (6)

(7) (8)

Figure 1.17: Assistive Feeding with right hand moving like a sinusoidal along z-axis around position $[0, 0.45, 0.4]$, deforming particularly the trajectory.

1.4 Example of application to human assistance task: Robotic Assistive Dressing

1.4.1 Definition of the problem

In this application, the aim is to assist users in one of the most difficult and critical phases of dressing: inserting the arm into the sleeve and pulling the sleeve towards the shoulder. This task is particularly important because

- It can potentially be dangerous, as users can become entangled in the sleeve and suffer discomfort or even injury
- It requires precision and careful handling of the sleeve

The arm is modeled as a curve $ABCD$ representing the axis of the arm. The piece of clothing is modeled as a rigid ring of area corresponding to the section of the shirt's sleeve. Therefore, the objective of the problem is to move, with a robotic manipulator, the ring along the arm axis from the tip of the arm to the shoulder, while maintaining the cross section of the ring perpendicular to the arm axis and while keeping the ring's center as close as possible to the arm axis [11].

This time, we don't have a repulsive field, but two attractive fields γ and β. γ's attractive point is the endpoint of the arm, i.e. the shoulder, where as for β, at every instant the attractive point is defined as the point T lying on the arm's axis, perpendicular projection of the center of the ring on the arm's axis.

Similarly as for the feeding case, the attractive field gamma is defined as a function of the parametric scalar s, defined in the interval[0;1] as:

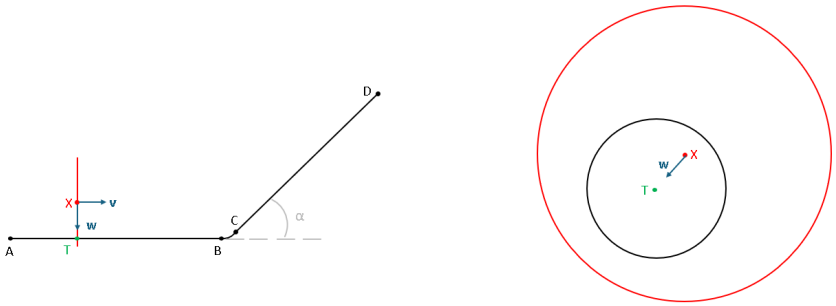
$$s = \frac{\|T - B\|}{length_{ABCD}} \tag{1.24}$$

The arm is modeled as a curve consisting of two straight lines representing the forearm and upper arm, and an arc representing the elbow connecting the lines. Thus, as we can see in Fig. 1.18 the curve is defined by 4 points: A in the tip of the arm, D in the shoulder, B and C are the connection between the lines and the arc. The lines are tangent to the arc in points B and C. The coordinates of the points of the curve depend on the input angle α between the forearm and the upper arm and change in a dynamic problem, but in all our configurations we start with point A placed in the origin and with the curve $ABCD$ on the plane x-z. For simplicity, I sized both the forearm and upper arm 30 cm long, thus in the elbow the value of s will be 0.5.

For collision detection between the ring and the arm, I sized the arm as a cylinder of radius 5 cm on the axis $ABCD$ and the rigid ring of radius 8 cm.

Trajectory definition

Referring again to Fig. 1.18, the point on which we will apply the control action is the center of the ring, called point X, which is connected rigidly to end effector. The end effector grabs the ring on the point of the ring placed along the positive y-axis. The task always starts with the point X on the plane y-z passing through point A. Point X will have to reach point D while being always as close as possible to the curve. The directions of the components of the control action are the normalized vectors $\mathbf{v}$, tangent to the arm's axis at every instant, and $\mathbf{w} = \frac{T-X}{\|T-X\|}$.



View in the x-z plane View in the y-z plane

Figure 1.18: Setting of the dressing problem. The black curve and the black circle represent the arm, where as the red ring is the rigid ring modeling the sleeve. The vectors $\mathbf{v}$ and$\mathbf{w}$ are the directions of the components of the control action.

1.4.2 Potential Fields in this application

This time there are no obstacles, but we want to avoid collision between the ring and the arm's surface. For this reason, only two potential fields are considered and they are both attractive fields.

Attractive field in point D

This is attractive field is necessary to complete the task. Its profile is given by the function γ and the direction of attraction is given by the

vector $\mathbf{v}$. Similarly as for the Feeding case, function γ will be a function of the parametric scalar s and will be shaped like a smoothed trapezoid to ensure gradual acceleration and deceleration phases. However, this time it is crucial, when the arm of the user is bent, that the manipulator slows down when approaching the elbow. For this reason, γ this time consists of two consecutive smoothed trapezoids. In fact, the velocity profile $\gamma(s)$ is divided into two halves: $\gamma_1(s)$ for $0 \leq s \leq 0.5$ and $\gamma_2(s)$ for $0.5 < s \leq 1$.

For $0 \leq s \leq 0.5$:

$$\gamma_1(s) = \begin{cases} \gamma_{min} + (\gamma_{max} - \gamma_{min}) \cdot \frac{1}{2} \left(1 - \cos\left(\frac{\pi s}{s_\mathrm{a}}\right)\right) & \text{if } s \in [0, s_1), \\ \gamma_{max} & \text{if } s \in [s_1, s_2), \\ \gamma_{max} + (\gamma_{min} - \gamma_{max}) \cdot \frac{1}{2} \left(1 - \cos\left(\frac{\pi(s - s_\mathrm{a} - s_\mathrm{c})}{s_\mathrm{d}}\right)\right) & \text{if } s \in [s_2, s_3). \end{cases} \tag{1.25}$$

For $0.5 < s \leq 1$:

$$\gamma_2(s) = \begin{cases} \gamma_{min} + (\gamma_{max} - \gamma_{min}) \cdot \frac{1}{2} \left(1 - \cos\left(\frac{\pi(s - 0.5)}{s_\mathrm{a}}\right)\right) & \text{if } s \in [s_3, s_4), \\ \gamma_{max} & \text{if } s \in [s_4, s_5), \\ \gamma_{max} + (\gamma_{min} - \gamma_{max}) \cdot \frac{1}{2} \left(1 - \cos\left(\frac{\pi(s - 0.5 - s_\mathrm{a} - s_\mathrm{c})}{s_\mathrm{d}}\right)\right) & \text{if } s \in [s_5, s_6]. \end{cases} \tag{1.26}$$

where, referring also to equation 1.24:

- s_a is the acceleration phase.
  $s_a = 0.05$
- s_c is the constant velocity phase.
  $s_c = 0.4$
- s_d is the deceleration phase phase.
  $s_d = 0.05$

- $s_1 = s_a$
  $s_2 = s_a + s_c$
  $s_3 = s_a + s_c + s_d$
  $s_4 = 2 \cdot s_a + s_c + s_d$
  $s_5 = 2 \cdot s_a + 2 \cdot s_c + s_d$
  $s_6 = 1$

The complete velocity profile $\gamma(s)$ is then defined by combining these two halves:

$$\gamma(s) = \begin{cases} \gamma_1(s) & \text{if } 0 \leq s \leq 0.5, \\ \gamma_2(s) & \text{if } 0.5 < s \leq 1. \end{cases} \tag{1.27}$$

Similarly to the feeding case, the parameters of the function γ, shown in Fig. 1.19, that need to be correctly designed are $\gamma_{min}, \gamma_{max}, s_d, s_c$, and s_a. For this application, a reasonable value for γ_{max}, considering user comfort and safety, is no more than 10 cm/s, while a suitable choice for γ_{min} is 1 cm/s.

Regarding the direction of the attractive velocity γ applied to point X, the vector $\mathbf{v}$ is always tangent to the arm. Specifically, for a given value of s, it is taken as the tangent vector to the curve at position s.

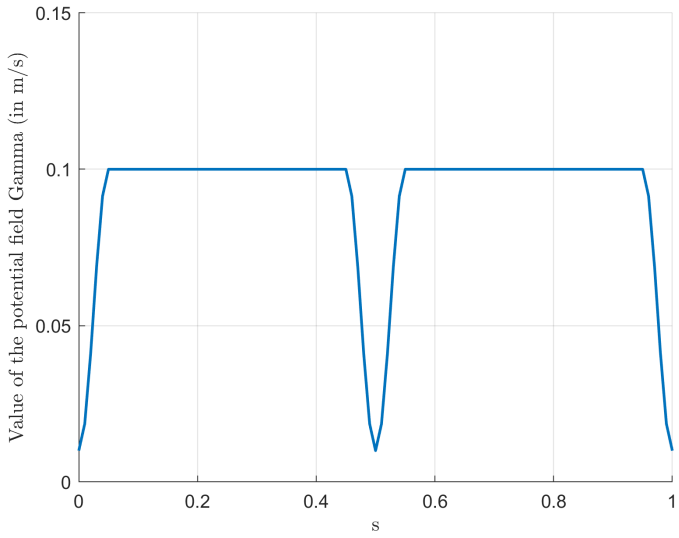


Figure 1.19: Profile of the attractive potential field $\gamma(s)$ in dressing task

Attractive field on the arm's axis

This attractive field is necessary to avoid collisions, and its profile is given by the function β. For the design of β, I tested two alternatives: the derivative of the sigmoid (β_1), mirrored with respect to the abscissa, and a parabola (β_1), as shown in Fig. 1.20. All functions are expressed in meters per second, with x in meters. Given that the ring radius is 6 cm and the arm radius is 3 cm, we want β to reach its maximum at $x = 8\,\text{cm} - 5\,\text{cm} = 3\,\text{cm} = 0.03\,\text{m}$.

$$\beta_1(x) = -4 \cdot \max_\beta \cdot \frac{\exp(-\theta_\beta \cdot x)}{(1 + \exp(-\theta_\beta \cdot x))^2} + \max_\beta \tag{1.28}$$

$$\beta_2(x) = \begin{cases} a \cdot x^2 & \text{if } 0 \leq x \leq 0.03, \\ a \cdot (0.03)^2 & \text{if } x > 0.03. \end{cases} \tag{1.29}$$

Where x is defined as:

$$x = \|T - X\| \tag{1.30}$$

I selected these shapes because both provide a gradual movement toward the arm's axis, with the value of the function being zero at the origin of the abscissa. β is a function of the distance between the ring's center and the point T on the arm's axis. The parameters max_β, θ_β, and a must be appropriately designed to ensure task success without collisions. For safety and comfort, it is reasonable to limit the maximum of these functions to a value no higher than 20 cm/s, to accommodate the potential movements of the user's arm. The values of θ_β and a must be such that β reaches its maximum when the ring is very close to the arm's surface.

For a specified maximum value of 20 cm/s, $max_\beta = 0.2\,\text{m/s}$ and an adequate value for θ_β would be 220, while a can be calculated as follows:

$$a \cdot (x = 0.03)^2 = 0.2 \tag{1.31}$$

$$a = \frac{0.2}{0.03^2} = 222.22 \tag{1.32}$$

With these values, the functions are shown in Fig. 1.20. We can observe that β_2 is wider than β_1, resulting in a more gradual change of speed, whereas β_1 is more responsive. Given the steep slope of the β_1 function, I made the reasonable choice of using the parabola profile for β, as it remains responsive while being more gradual.

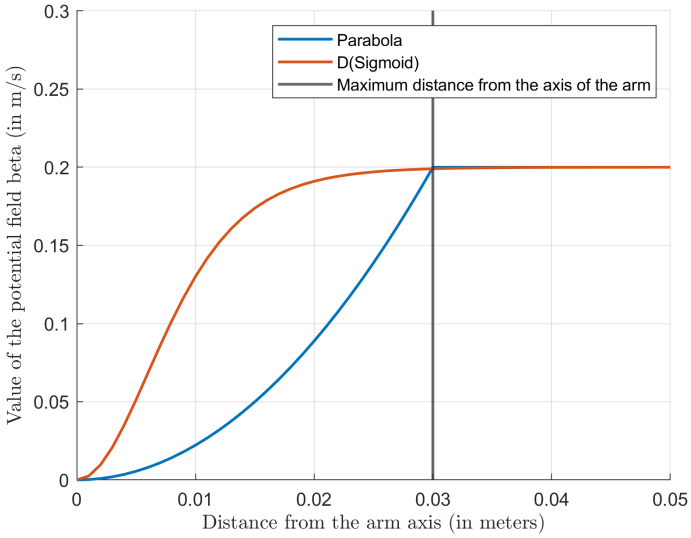


Figure 1.20: Comparison of the possible shapes (parabola or derivative of the sigmoid) of the attractive potential field β in dressing task. The parabola is more gradual and always increases in steepness, it is the better profile for this application.

1.4.3 Control Loop

In the case of Dressing, the controller is described with the following equations:

The control input $\mathbf{u}_k$ is defined as:

$$\mathbf{u}_k = \mathbf{v}_k \cdot \gamma(s_k) + \mathbf{w}_k \cdot \beta(\|X_k - T_k\|) \tag{1.33}$$

The position update X_{k+1} at instant $(k+1)$ is given by:

$$X_{k+1} = X_k + \mathbf{u}_k \cdot dt \tag{1.34}$$

where:

- $\mathbf{v}_k$ is the direction of application of the attractive field tangent to the arm's axis at instant k
- $\mathbf{w}_k$ is the direction of application of the attraction on the end effector towards the arm's axis at instant k
- X_k is the position of the manipulator's end effector at the instant k
- T_k is the orthogonal projection of X_k on the arm's axis at the instant k

- $\mathbf{u}_k$ is the control action at the instant k

Moreover, the orientation of the end effector is updated at every instant k in such a way that the ring's surface is always perpendicular to the axis of the arm. This is done by computing the rotation matrix, $\mathbf{R}_k$, which aligns the x-axis with the vector $\mathbf{v}_k$. To do this, we have to find the axis of rotation and the angle required to rotate the unit vector along the x-axis, $(1, 0, 0)$, to the vector $\mathbf{v}_k$.

First, we determine the axis of rotation by taking the cross product of the vectors $(1, 0, 0)$ and $\mathbf{v}$. The axis of rotation, $\mathbf{r}$, is given by:

$$\mathbf{r} = \frac{\mathbf{v} \times (1, 0, 0)}{\|\mathbf{v} \times (1, 0, 0)\|} \tag{1.35}$$

Then, we compute the angle, θ, between $(1, 0, 0)$ and $\mathbf{v}$ using the dot product:

$$\theta = \arccos\left(\frac{\mathbf{v} \cdot (1, 0, 0)}{\|\mathbf{v}\| \cdot \|(1, 0, 0)\|}\right) \tag{1.36}$$

The rotation matrix $\mathbf{R}$ that rotates a vector by θ around the axis $\mathbf{r}$ can then be constructed using Rodrigues' rotation formula [12]:

$$\mathbf{R} = \mathbf{I} + \sin(\theta)\mathbf{K} + (1 - \cos(\theta))\mathbf{K}^2 \tag{1.37}$$

where $\mathbf{K}$ is the skew-symmetric matrix corresponding to the axis $\mathbf{r}$.

1.4.4 Performance Analysis

As we did for the feeding application, we will analyze the performance of our solution using nearly the same effectiveness parameters. We will use the same sample time dt of 0.1 seconds.

- Regarding accuracy, the situation here differs from feeding. In an application like dressing, the maximum allowed distance between the end-effector's final position and the endpoint (point D) should be in the order of millimeters. However, while accuracy in the feeding task was controlled by the input parameter b in the iteration loop, for dressing, the iteration loop is governed by the condition "while $s < 1$." This means that in dressing, the priority is that point T reaches the shoulder, while point X only needs to be in a position on the plane that avoids collisions. Hence, accuracy is determined by the distance between point T and point D. Due to the absence of repulsive fields acting tangentially to the arm's axis, the accuracy will be very high, with values less than one millimeter. Therefore, accuracy in our ideal

simulations will not be taken into consideration, as $\|D - T\|$ in the absence of obstacles will never exceed $dt \cdot \gamma_{min} = 1\,\text{mm}$.

- Time efficiency is simply measured by the task completion time and compared to the baseline case.

1.4.5 Static Dressing, User Remains Completely Still

In this scenario, considered the baseline for performance analysis, the user's arm remains still throughout the entire dressing operation. Ideally, the user would be asked to position their arm in a straight line, but we will test the algorithm for various elbow angles α. From now on, we will test the solution using the designed functions γ and β, as previously described. To test the reliability of the function β for a static arm, the starting position of the end-effector is placed at coordinates $X = A + [0; 0.029; 0]$, which positions the center very close to the maximum allowed distance from the axis.

1. Straight arm ($\alpha = 0°$)
2. Arm bent at 45 degrees ($\alpha = 45°$)
3. Arm bent at 90 degrees ($\alpha = 90°$)

	0°	**45°**	**90°**
Success	yes	yes	yes
Time of Completion [s]	**8.7**	9.1	9.7
Collision Between Ring and Arm	no	no	no

Table 1.7: Results of the simulation for Static Assistive Feeding

The results of the simulations are shown in Table 1.7. In all three cases, the system successfully completes the task without any collisions. The distance $\|D - T\|$ consistently remains below 4 millimeters.

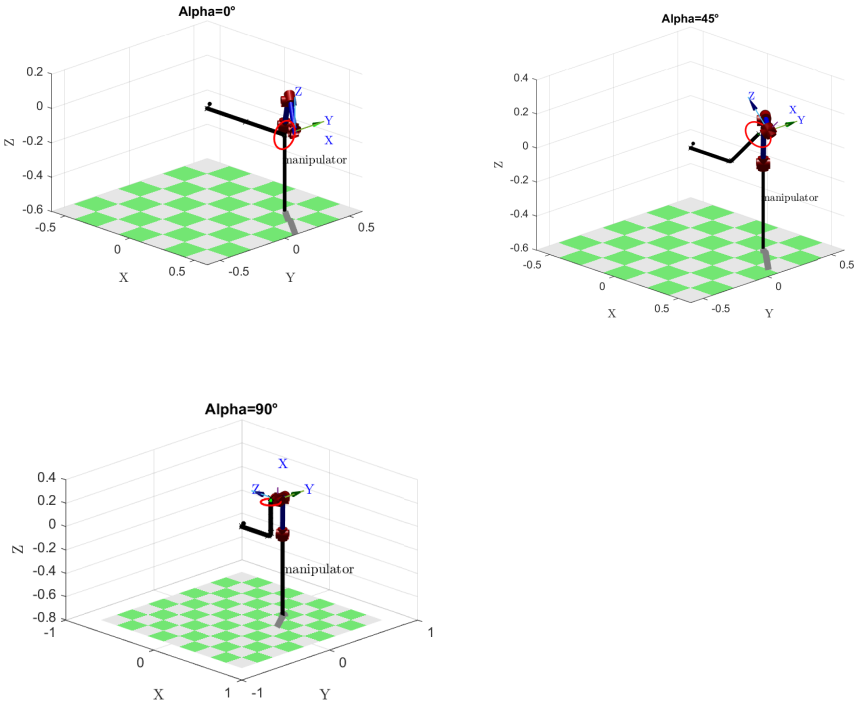
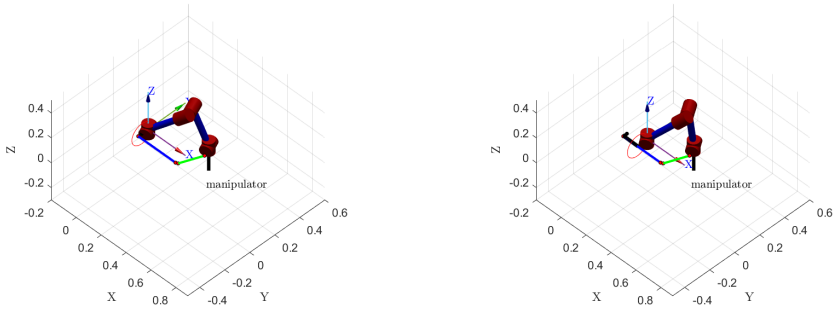
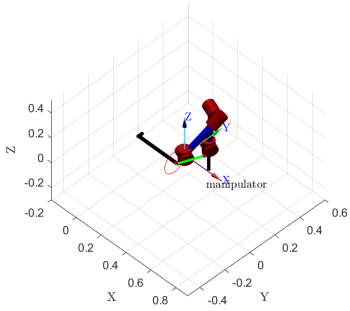


Figure 1.21: Simulation of static assistive dressing task with various elbow angles. The images are the last frame of the animation. The black dots represent the positions taken by the center X of the rigid ring in red. Point X starts at the tip of the arm with an offset of 2.9 cm from the axis of the arm.

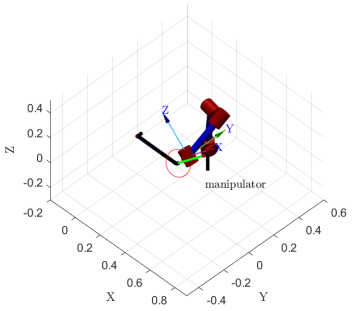


(1)

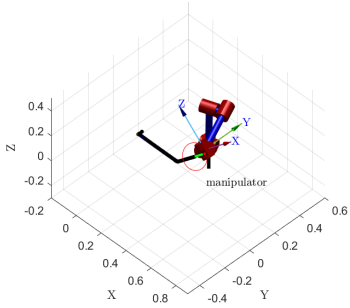
(2)



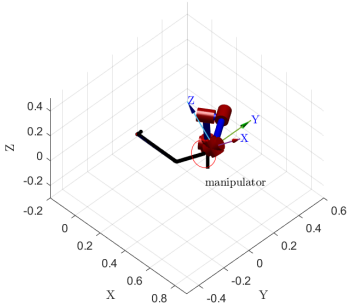
(3)



(4)



(5)



(6)

Figure 1.22: Static Assistive Dressing, sequence of snapshots of the simulation with $\alpha = 45°$.

1.4.6 Dynamic dressing, user moving the arm

To test our algorithm in a dynamic environment, we will consider the user's arm movement. In a real situation, the user might involuntarily move their arm while the manipulator is operating but still expect the task to be completed. Thus, I decided to test the system on an oscillating rotation of the arm around the shoulder, as under the assumption of a fixed torso and head, the shoulder remains stationary. The angle of rotation of the arm, ξ, follows a sinusoidal function. The angle and angular velocity with respect to time are expressed as:

$$\xi(t) = \xi_{\max} \cdot \sin(a \cdot t) \tag{1.38}$$

$$\dot{\xi}(t) = a \cdot \xi_{\max} \cdot \cos(a \cdot t) \tag{1.39}$$

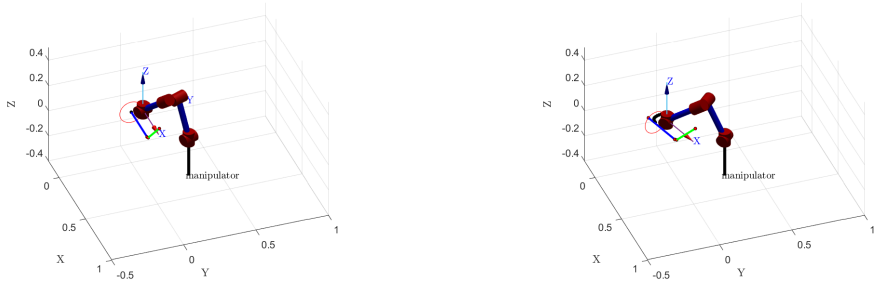
I tested the algorithm with the following parameters:

- Starting position of the center of the ring, X_0, at A
- $\alpha = 45°$
- $\xi_{\max} = 15°$, resulting in a total arm rotation range of 30 degrees
- Increasing values of a until collision. The values of a considered are [0.5, 1, 1.39, 1.5]. The value $a = 1.39$ is the limit at which the algorithm still functions, given a sample time of 0.1 seconds. With smaller sample times, a higher value of a can be tolerated.

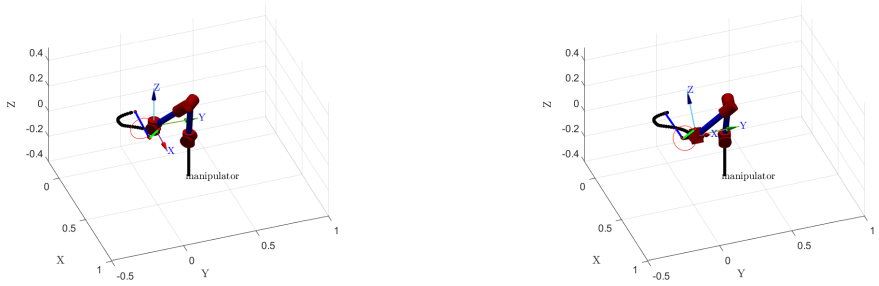
$\alpha = 45°, dt = 0.1s$	$a = 0$	$a = 0.5$	$a = 1$	$a = 1.39$	$a = 1.5$
Success	yes	yes	yes	yes	no
Time of Completion [s]	9.1	8.9	8.7	**8.5**	-
Collision Ring-Arm	no	no	no	no	yes

Table 1.8: Results of the simulations for Dynamic Assistive Feeding with various values of the scalar a

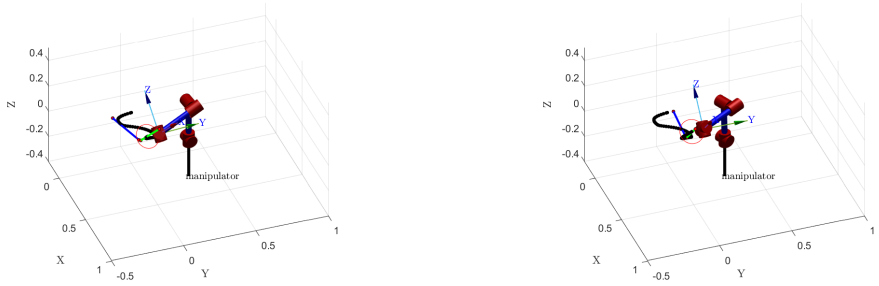
We can see in table 1.8 the results of the simulations and we show in Fig. 1.23 a sequence of the animation with $a = 1.39$.



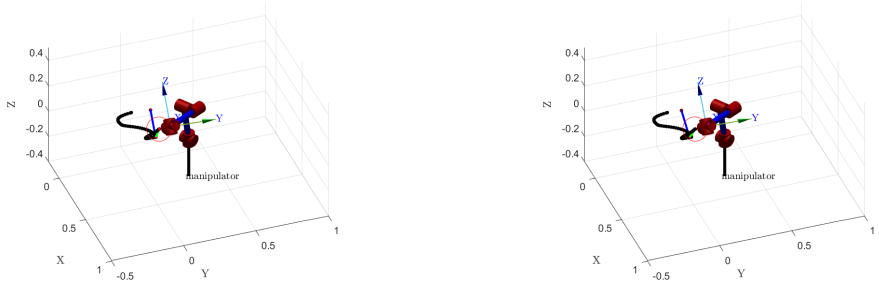
(1) (2)



(3) (4)



(5) (6)



(5) (6)

Figure 1.23: Sequence of snapshots of the simulation of dynamic assistive dressing with $\alpha = 45°$, $\Delta t = 0.1$ s and $a = 1.39$. The movement of the arm obliges the manipulator to engage a complex trajectory, but succeeds in keeping its distance from the arm in the limits.

1.5 Considerations for possible future real implementations

Limitations of the simulation setup

- Dynamics not considered:
  The simulation environment does not account for the dynamics of the robotic manipulator or the objects it interacts with. Without dynamics, the system may exhibit unrealistic responses when applied to real-world scenarios. For example, inertia, friction, and gravity, which influence how forces are applied and how the robot responds, are ignored.

- Noise and measurement precision:
  The simulations rely on idealized measurements without accounting for noise. In a real system, sensor noise (especially in RGB-D cameras) and the precision of the robotic arm could lead to degraded performance. However, we will tackle the effect of noise in the section 2.1.5.

- Idealized sampling time:
  The sample time chosen in simulations (0.1s) may not be adequate for real-world applications, where faster processing might be necessary to capture rapid movements and avoid accidents.

Requirements for a physical setup

- Inclusion of system dynamics:
  The real-world dynamics of the robotic arm, including friction, inertia, and forces, need to be modeled accurately. This requires an understanding of the physical properties of the manipulator and the environment.

- Robotic manipulator:
  A high-precision robotic arm (for example a UR5) with a payload capacity matching the tasks could be used. This system must have reliable torque control to handle forces and moments when interacting with objects or assisting humans.

- Sensors:
  RGB-D cameras for depth perception and accurate mapping of the environment are required. However, these sensors have inherent limitations in accuracy, particularly for fine movements, which should be considered when comparing simulation to real-world performance.

Performance analysis in real setup

- Accuracy:
  In a real setup, the manipulator's accuracy will primarily depend on the precision of the sensors (e.g., RGB-D cameras) and actuators. While robotic arms can reach millimeter-level precision, RGB-D cameras can reach lower accuracy levels, which could impact the end-effector's final positioning. To achieve maximum accuracy, fine-tuning of the control algorithm with appropriate feedback from both the manipulator and sensors is crucial.

- Actuator precision:
  The robot's servos or actuators are typically very precise (in the millimeter range), but joint backlash or mechanical imperfections could introduce small errors in a real setup.

- Calibration of camera and manipulator:
  The calibration between the camera and the manipulator is needed. Any misalignment will directly affect accuracy.

- Time of completion of the task:
  We obtained low ideal task completion times in our simulations. For example, the completion time was of around 10 seconds for the baseline feeding scenario without obstacles. In real-world studies, task completion times for robotic assistive feeding vary depending on system complexity and environmental factors. For example, in the article [13], the task took about 30 to 60 seconds per feeding interaction due to more precise movements and handling diverse food textures. This real-world time accounts for the complexities involved in food manipulation, user interaction, and safety protocols.

Example of possible real setup and approximate costs

- Robotic manipulator:
  UR5 robotic arm (30,000 - 50,000) euros [14]

- RGB-D Camera:
  Intel RealSense D435i (300 - 500) euros [15]

- Calibration tools:
  Laser trackers or high-precision measurement devices (5,000 - 10,000) euros [16]

- Additional sensors (Force/Torque Sensors):
  To measure the interaction forces between the robot and the environment (3,000 - 5,000) euros [17]

2. Active User and State Space Control

After addressing motion control for a manipulator using potential fields with a passive user, we will now explore two examples of active motion control of the manipulator by the user. The first case involves motion control through direct imitation, using state prediction and noise filtering with the Kalman Filter. This approach is interesting as it deals with noise and enables real-time imitation of user movements. The second case involves motion control with Model Predictive Control (MPC) for real-time tracking of the user's motion. This is notable because it also considers the inertia of the object being moved, as an example we consider the case of transporting a glass of water. Moreover, this system has two coexisting objectives: following a reference trajectory in real-time for the glass and avoiding water sloshing.

2.1 Motion Control by Direct Imitation Considering the Effect of Noise and Without Considering Inertia: Application to Box Collaborative Transport

In this problem, I will simulate the collaborative lifting of a heavy box involving a human and a manipulator that imitates the human's actions while applying the necessary force. This task is useful for individuals who have difficulty lifting heavy objects. Controlling the manipulator via teleoperation is an effective solution, allowing the user to lift and place the heavy object without exertion. The manipulator will copy the movements of the user's hand after an initial calibration step, during which the offset between the hand and the end-effector is established and maintained.

Key Considerations:

- **Safety**: Ensuring the manipulator operates without causing harm.
- **Imitation Accuracy**: The manipulator should precisely imitate the user's movements for smooth and coordinated lifting.

- **Real-time Response**: The system must respond in real-time to the user's actions to maintain natural interaction and prevent accidents.

In this study, we assume all preliminary setup actions are complete. We will not consider the manipulator's dynamics or gripping mechanics but instead focus on designing the control logic necessary to complete the task. The forces and moments required from the manipulator will be addressed in future work.

2.1.1 Problem setup

I will simulate a simplified scenario where the manipulator's base is fixed, and the user aims to move a box. The manipulator grips and fully supports the box from the center of one face, while the user's hand is on the opposite side with an offset equal to the length of the box's side. As in section 1.3, the hand is modeled as a cube with a side length of 10 cm, and the end-effector copies its position. The box, modeled as a cube with a side length of 20 cm, is lifted and displaced by the user. Since we are working in a simulation environment, we define the hand's trajectory and assume new measurements are available at each time step. The trajectory of the hand is defined as lifting and moving the box from the ground to an elevated surface. It consists of:

- A linear displacement along the z-axis of 0.7 meters to lift the box from the ground at a constant speed $v = 0.1\,\mathrm{m/s}$,
- A linear displacement along the x-axis of 0.5 meters to position the box at the desired location, at speed v,
- A linear displacement along the negative z-axis of 0.1 meters to place the box on the new surface at speed v.

The trajectory of the hand's center is defined on the x-z plane and is shown in Fig. 2.1.

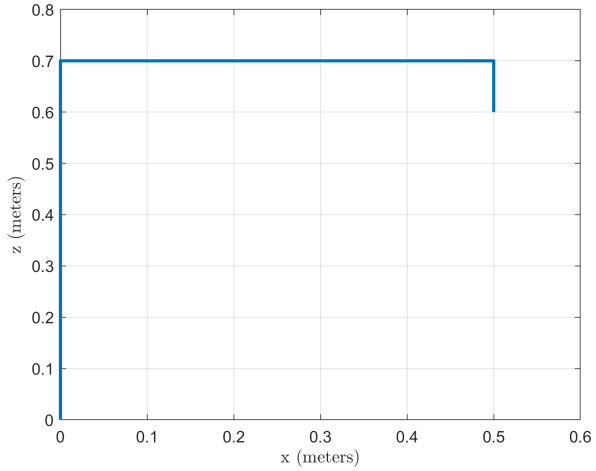


Figure 2.1: Trajectory of the user's hand

The manipulator is a collaborative robot with its base at $[0.25, 0.55, 0]$, and the Denavit-Hartenberg (D-H) parameters are listed in Table 2.1.

Joints	**Theta**	**d [m]**	**a [m]**	**Alpha [rad]**	**Offset [m]**
1	q_1	0	0	$\frac{\pi}{2}$	0
2	q_2	0	0.5	0	0
3	q_3	0.01	0.0203	$-\frac{\pi}{2}$	0
4	q_4	0.5	0	$\frac{\pi}{2}$	0
5	q_5	0	0	$-\frac{\pi}{2}$	0
6	q_6	0	0	0	0

Table 2.1: D-H Parameters for the Robotic Manipulator in Box Collaborative Transport

2.1.2 Performance

In this ideal case of direct imitation, without considering the inertia of the box, the simulations will show ideal performance. The key performance considerations are:

1. **Sample time Δt**:
   As discussed in section 1.3, sample time is a crucial parameter affecting system performance. Here, we will also use a sample time of 0.1 seconds.

2. **Effect of noise**:
   We will introduce white noise into the measurement of the user's hand

position and test the system's response after applying a suitably tuned Kalman Filter.

3. **Effect of lost measurements**:
   It is possible that the hand position measurements may occasionally be unavailable. The Kalman Filter allows the system to estimate the state for a certain number of time steps without measurements.

4. **Real-time response for video-frequencies from 30 to 60 fps**:
   Using the Kalman Filter will enable state prediction at the next instant, allowing for a simultaneous response of the system.

I will begin by addressing the simplest case: direct imitation without state prediction and estimation.

2.1.3 Direct imitation without prediction of the state

It is an operation of collaborative lifting and positioning of heavy objects like a heavy box with direct imitation of the user hand motion at every instant. For the moment, we are only considering the position and not the orientation.

Control Loop

$$\mathbf{x_{EE}}_{(0)} = \mathbf{x}_{(0)} + \mathbf{o} \tag{2.1}$$

$$\dot{\mathbf{x}}_{\mathbf{EE}(0)} = \dot{\mathbf{x}}_{(0)} + \mathbf{o} \tag{2.2}$$

$$\mathbf{x_{EE}}_{(k)} = \mathbf{x}_{(k-1)} + \mathbf{o} \tag{2.3}$$

$$\mathbf{x}\dot{\mathbf{E}}\mathbf{E}_{(k)} = \dot{\mathbf{x}}_{(k-1)} + \mathbf{o} \tag{2.4}$$

where

- $\mathbf{x_{EE}}_{(k)}$ and $\mathbf{x}\dot{\mathbf{E}}\mathbf{E}_{(k)}$ are the position and the speed vectors of the end effector at time step k
- $\mathbf{x}_{(k-1)}$ and $\dot{\mathbf{x}}_{(k-1)}$ are the position and speed of the hand at time step $(k-1)$

$$\mathbf{x}_{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \; ; \dot{\mathbf{x}}_{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.5}$$

- $\mathbf{o}$ is the constant offset vector between the end effector and the center of the hand.

$$\mathbf{o} = \begin{bmatrix} 0 \\ l_b + 0.5 \cdot l_h \end{bmatrix}$$

where l_b and l_h are the side lengths of the box and of the hand, both modeled as cubes

In this ideal configuration, the simulations perform as expected, with the end-effector's trajectory perfectly copying the hand's trajectory. However, this preliminary system has two key issues:

- **Lack of simultaneous direct imitation**: At each time step k, a new measurement of the hand's position $\mathbf{x}_{(k)}$ is taken, while the end-effector is still in the position $\mathbf{x}_{(k-1)}$. The end-effector will reach $\mathbf{x}_{(k)}$ only at the next time step $k+1$.
- **Inability to handle noise in the hand's movement**: The end-effector attempts to perfectly replicate every movement of the hand, including unwanted or erroneous movements. This is particularly evident in cases of noise, such as when modeling a user with motor impairments, which may result in unintended movements. Here, when I refer to noise, I am not referring to measurement noise, as this is a simulation environment with "perfect" measurements, but rather to unwanted variations in the user's actual movements.

2.1.4 Direct imitation with Kalman Filter for the prediction and estimation of the state at every instant

Dealing with the first issue stated in the previous case, the inability to imitate the hand trajectory in real-time, the solution I propose is to apply a Kalman filter for the prediction and estimation of the state at each instant [18]. This new control logic allows the system to predict and estimate the hand's state at the next instant, leading to synchronous motion of the hand and the end-effector.

Control Loop:

The hand state at step k is defined as:

$$x_k = \begin{bmatrix} \mathbf{x}_{(k)} \\ \dot{\mathbf{x}}_{(k)} \end{bmatrix} \tag{2.6}$$

The hand state transition matrix is:

$$A = \begin{bmatrix} 1 & 0 & 0 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 & dt & 0 \\ 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

Kalman Filter Initialization Step:

$$\hat{x}_{0|0} = \text{Initial state estimate} = z_0 \tag{2.8}$$

$$P_{0|0} = \text{Initial covariance estimate} \tag{2.9}$$

Kalman Filter Prediction Step:

$$\hat{x}_{k|k-1} = A_k \hat{x}_{k-1|k-1} \tag{2.10}$$

$$P_{k|k-1} = A_k P_{k-1|k-1} A_k^T + Q_k \tag{2.11}$$

where:

- $\hat{x}_{k|k-1}$ is the predicted state estimate at time step k,
- A_k is the state transition matrix (constant for a linear time-invariant system): $A_k = A$,
- $\hat{x}_{k-1|k-1}$ is the previous state estimate,
- $P_{k|k-1}$ is the predicted covariance matrix,
- $P_{k-1|k-1}$ is the previous covariance estimate,

- Q_k is the process noise covariance (model uncertainty), which is constant in this case.

Kalman Filter Update Step:

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} \tag{2.12}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - H_k \hat{x}_{k|k-1}) \tag{2.13}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{2.14}$$

where:

- K_k is the Kalman gain at time step k,
- H_k is the observation matrix (maps the state to the measurement space), which is constant,
- R_k is the measurement noise covariance matrix, also constant,
- $\hat{x}_{k|k}$ is the updated state estimate after incorporating the measurement,
- z_k is the actual measurement at time step k,
- $H_k \hat{x}_{k|k-1}$ is the predicted measurement,
- $P_{k|k}$ is the updated covariance estimate,
- I is the identity matrix.

End-effector update:

$$x_{EE(k)} = \hat{x}_{k|k} + \mathbf{o} \tag{2.15}$$

If no measurements are available, we cannot perform the update step, and we will use the prediction step only:

$$x_{EE(k)} = \hat{x}_{k|k-1} + \mathbf{o} \tag{2.16}$$

where $x_{EE(k)}$ is the state of the end-effector at time step k.

Tuning of the Matrices

The goal is to achieve a balance between responsiveness, stability, and noise suppression.

1. **Observation Matrix** (H_k):
   This matrix maps the system's state to the measurement space, relating the state vector to the observed measurements. Since we are measuring all parts of the hand's state vector:

$$H_k = H = I \tag{2.17}$$

2. **Process Noise Covariance Matrix** (Q_k):
   This matrix represents uncertainty in the system model, capturing how much the actual process deviates from the assumed model. Q_k is tuned by trial and error. Larger values account for more uncertainty in the model, causing the filter to rely more on measurements. In this case, since there is no noise:

$$Q_k = Q = 0 \cdot I \tag{2.18}$$

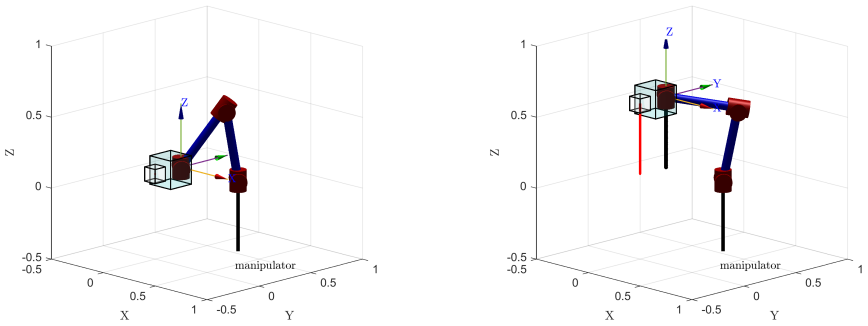
3. **Measurement Noise Covariance Matrix** (R_k):
   This matrix represents uncertainty in the measurements, capturing sensor noise characteristics. Since there is no noise in this simulation:

$$R_k = R = 0 \cdot I \tag{2.19}$$

4. **Initial Covariance Matrix** (P_0):
   This matrix represents initial uncertainty in the state estimate. If there is little prior information, we set the diagonal elements of P_0 to large values. In our ideal case:

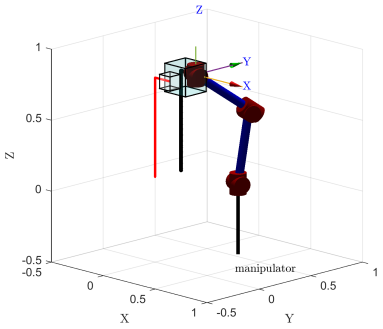
$$P_0 = I \tag{2.20}$$

After selecting the Kalman filter matrices, we test the system. For 10 time instants along the horizontal segment of the trajectory, the system will have no measurements of the hand. Thanks to the prediction step of the Kalman filter, the manipulator will update the end-effector position using the model, as described in equation 2.10. The results are shown in Fig. 2.2. In the figures, the red trajectory represents the trajectory of the hand's center as seen by the sensors, while the black trajectory represents the end-effector's path. In the fourth image, the red trajectory disappears, indicating missing measurements, yet the end-effector continues along its predicted trajectory. After 10 time instants, the measurements resume, and the system successfully completes the task without errors.

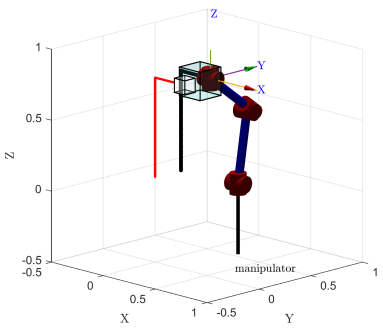


(1)

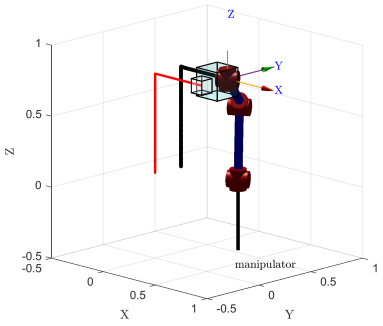
(2)



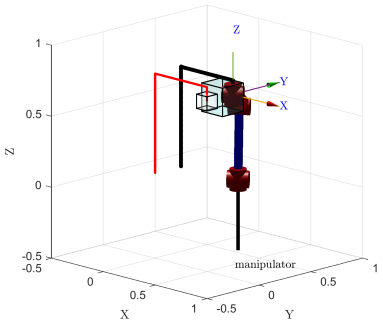
(3)



(4)



(5)



(6)

Figure 2.2: Sequence of snapshots of the simulation of Box collaborative Transport with Kalman Filter and lack of hand measurements for 10 time instants on the horziontal part of the trajectory. The end effector follows perfectly the reference trajectory.

2.1.5 Modeling Motor Impairment as White Noise in Human Hand Movement: Direct Imitation Using a Kalman Filter for State Prediction and Noise Filtering

Addressing the second issue mentioned in the first case, namely noise in the hand's movement due to a motor impairment such as Parkinson's disease, the Kalman Filter again offers an effective solution. By filtering out undesired movements and retaining only the relevant ones, the Kalman Filter helps ensure smoother motion. The amount of trajectory information that is filtered can be controlled by proper tuning of the matrices.

Model of the Noise in Hand Motion

The noise is modeled as a random component added to the hand's position and velocity, specifically as white noise. White noise is a random signal with a flat power spectral density, meaning it has equal intensity across different frequencies. Mathematically, white noise can be modeled as a random process where each sample is drawn from a normal (Gaussian) distribution with a mean of zero and positive variance:

$$\mathbf{n}(k) = \sigma \cdot Z(k) \tag{2.21}$$

where:

- $\mathbf{n}(k)$ is the value of the white noise at time step k,
- σ is the standard deviation (spread or intensity of the noise). In this application, I chose $\sigma = 0.01$ meters (1 cm),
- $Z(k)$ is a random variable drawn from a normal distribution $N(0, 1)$, meaning a Gaussian distribution with a mean of 0 and variance of 1.

Thus, the measurement of the state at step k will be:

$$z_k = \begin{bmatrix} \mathbf{x}_k + \mathbf{n}_k \\ \dot{\mathbf{x}}_k + \mathbf{n}_k \end{bmatrix} \tag{2.22}$$

Compared to the previous case, we need to properly tune the matrices Q and R to achieve the best trajectory for the manipulator. Since there is no uncertainty in our model and we are dealing with noise in the measurements, we want the system to rely more on the model than the measurements. Due to the noise, the hand will follow a rough and complex trajectory, but we want the end-effector to move as smoothly as possible, filtering out the noise.

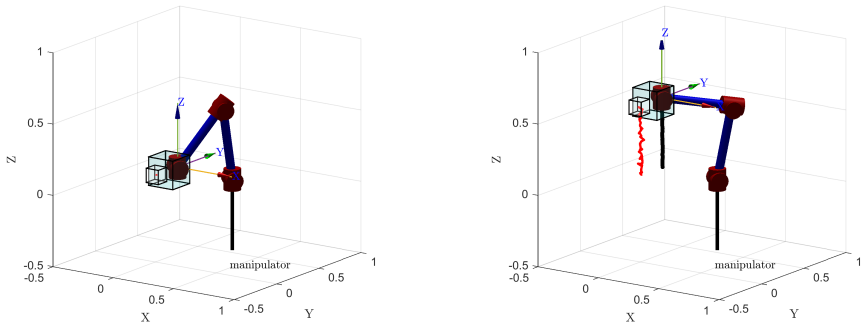
- **Process Noise Covariance Matrix** (Q)
  Since there is no uncertainty in the model, we want a small value for Q. I will test different decreasing values of Q, from 1 to 0.01 times the identity matrix.

- **Measurement Noise Covariance Matrix** (R)
  Since we are in a simulation environment without sensor specifications, we will determine R experimentally. We aim to filter the noise and rely more on the model, so we will use large values for R. I will test different values of R, ranging from 1 to 100 times the identity matrix.

Observing the results of the simulations, we obtain the smoothest trajectory for the manipulator with the following combinations:

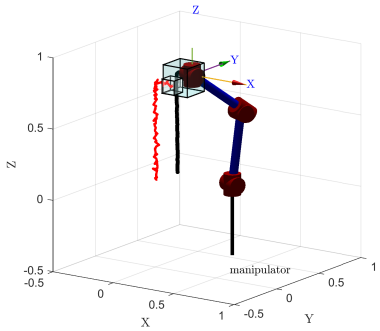
- $Q = 0.01 \cdot I$; $R = 1 \cdot I$,
- $Q = 0.1 \cdot I$; $R = 10 \cdot I$,
- $Q = 1 \cdot I$; $R = 100 \cdot I$.

If we choose lower values for Q and higher values for R, the manipulator will not follow the desired trajectory. For example, using the combination $Q = 0.1 \cdot I$; $R = 10 \cdot I$, we show the animation sequence in Fig. 2.3, where the noisy motion of the hand is visible, while the manipulator's motion remains smooth. In this case, for 10 iterations along the horizontal part of the trajectory, there are no hand state measurements. In Fig. 2.4, we present the result of the simulation with excessive reliance on the model, i.e., very low Q and very high R, specifically $Q = 0.01 \cdot I$; $R = 100 \cdot I$.

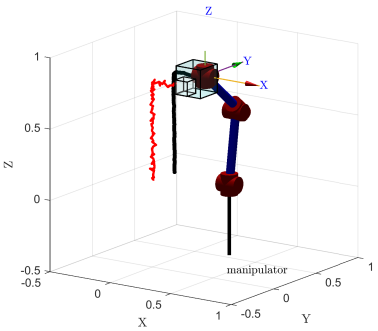


(1)

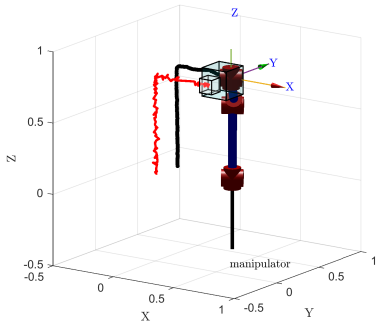
(2)



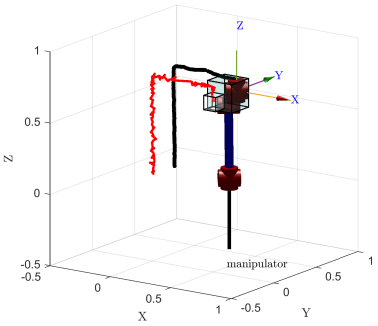
(3)



(4)



(5)



(6)

Figure 2.3: Sequence of snapshots of the simulation of box collaborative transport with the presence of white noise on the hand motion to simulate motion impairment of the user and with $Q = 0.1 \cdot I$; $R = 10 \cdot I$

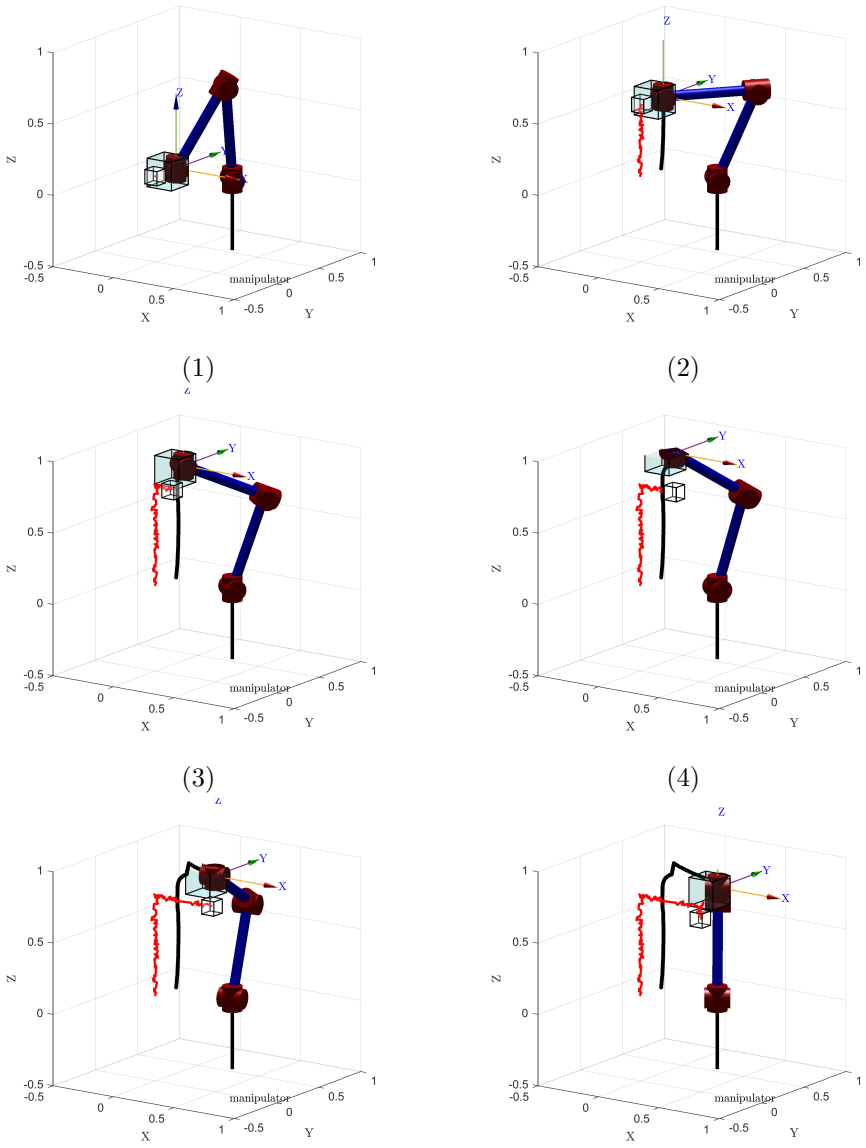


Figure 2.4: Sequence of snapshots of the simulation of box collaborative transport with the presence of white noise on the hand motion to simulate motion impairment of the user and with bad choice of the Q and R matrices: $Q = 0.01 \cdot I$; $R = 100 \cdot I$. When measurements are not available, the robotic manipulator diverges from the reference.

2.2 Motion control with consideration of the inertia of the system: Model Predictive Control trajectory tracking for direct imitation

To complicate the problem, we will now consider a case of direct imitation of the user where we take in consideration the inertia of the object we want to move. A good example of this scenario in the field of robotic assistance is for example the motion control of a glass of water, where the system controls the position and speed of the glass of water, while avoiding the water overflow from the glass. This being a real-time multi-objective control with constraints, a suitable controller for this problem could be a Constrained Trajectory Tracking Model Predictive Control.

2.2.1 Definition of the problem and 2-DOFs model of the system

This problem was modeled in 2D with a cart and a pendulum centered in the cart moving only on one axis (x). The reason of this simplification is that the relevant motion making the water spill from the glass is only the horizontal motion. Thus, the system has two degrees of freedom: the motion of the cart and the oscillations of the pendulum. The simple pendulum models the sloshing of the water inside of the glass; the main assumption here is that the liquid free surface always remains planar, so that its oscillation can be modeled as a pendulum attached to the line normal to the liquid plane [19]. The 2D model is shown in Fig. 2.5.

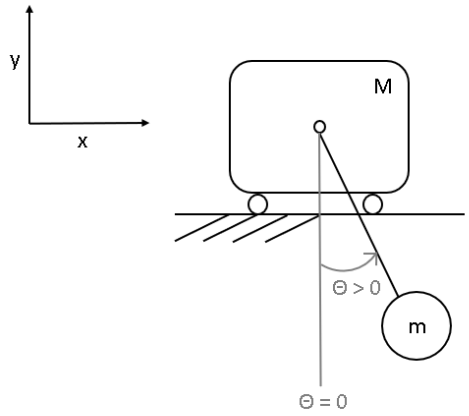


Figure 2.5: 2D model of the problem cart (of mass M) and pendulum (of mass m). The pendulum length is l.

For this application I referred to a cylindrical glass 15 cm tall, 7 cm of diameter and filled up with water until 13 cm. Thus, I chose the following values of the system's parameters to model the glass of water:

$$h_g = 0.15 \text{ m} \tag{2.23}$$
$$D_g = 0.07 \text{ m} \tag{2.24}$$
$$l = 0.13 \text{ m} \tag{2.25}$$
$$m = \rho_w \cdot l \cdot \pi \cdot \frac{D_g^2}{2} = 0.5 \text{ kg} \tag{2.26}$$
$$M = 0.25 \text{ kg} \tag{2.27}$$
$$\zeta = 0.005 \tag{2.28}$$

where:

- h_g is the glass height.
- D_g is the glass diameter.
- l is the length of the pendulum (height of the water column).
- m is the pendulum's mass (mass of the water column).
- M is the mass of the cart (mass of the glass).
- ζ is the damping ratio of the pendulum's oscillations. This value was calculated using the damping formulation for a liquid in a circular container found in article [19], that models the sloshing of a liquid in a circular container.

From the geometry of the system we can calculate the maximum angle between the water surface and the horizontal to avoid water spilling: this angle will be called θ_{max}. That will be the maximum angle of the pendulum with respect to the vertical in our cart and pendulum simple 2D model.

$$\theta_{max} = \text{atan}(\frac{h_g - l}{0.5 \cdot D_g}) = 0.5191 \text{ rad} = 29.74^\circ \tag{2.29}$$

Complicating the system by moving a tray with a glass of water on top could be solved just by imposing a constraint for the maximum value of the force (control action). That upper limit would be the force that makes the glass slip on the tray, it depends on the friction coefficient between the glass and the tray. We assume that friction is high enough so that the glass does not slide on the tray.

Model of the system

Now, we will derive the equations of motion and the state space representation of the 2D system cart and pendulum. First, through the Lagrange Equation I obtained the equations of motion of the 2-DOFs system, considering a damping in the oscillations of the pendulum's angle, to simulate the damping in the water oscillations inside the glass.

Hypotheses:

- Damping due to friction between cart and surface it's moving on is null.
- ζ is the damping ratio for the pendulum's oscillations.

Forces acting on the system: Horizontal force $F(t)$ acting on the cart.

Kinetic Energy:

$$T = \frac{1}{2}M\dot{x}_c^2 + \frac{1}{2}m(\dot{x}_p^2 + \dot{y}_p^2) \tag{2.30}$$

Potential Energy:

$$V = mg(-l\cos\theta) \tag{2.31}$$

Coordinates:

$$x_p = x_c + l\sin\theta \tag{2.32}$$
$$y_p = y_c - l = -l\cos\theta \tag{2.33}$$
$$\dot{x}_p = \dot{x}_c + l\dot{\theta}\cos\theta \tag{2.34}$$
$$\dot{y}_p = l\dot{\theta}\sin\theta \tag{2.35}$$

Substituting into Kinetic Energy equation:

$$\begin{aligned} T &= \frac{1}{2}M\dot{x}_c^2 + \frac{1}{2}m((\dot{x}_c + l\dot{\theta}\cos\theta)^2 + (l\dot{\theta}\sin\theta)^2) \\ &= \frac{1}{2}M\dot{x}_c^2 + \frac{1}{2}m(\dot{x}_c^2 + 2l\dot{x}_c\dot{\theta}\cos\theta + l^2\dot{\theta}^2\cos^2\theta + l^2\dot{\theta}^2\sin^2\theta) \\ &= \frac{1}{2}M\dot{x}_c^2 + \frac{1}{2}m\dot{x}_c^2 + ml\dot{x}_c\dot{\theta}\cos\theta + \frac{1}{2}ml^2\dot{\theta}^2 \end{aligned} \tag{2.36}$$

Lagrangian $L = T - V$:

$$\begin{aligned} L &= \frac{1}{2}(M+m)\dot{x}_c^2 + ml\dot{x}_c\dot{\theta}\cos\theta + \frac{1}{2}ml^2\dot{\theta}^2 - mg(-l\cos\theta) \\ &= \frac{1}{2}(M+m)\dot{x}_c^2 + ml\dot{x}_c\dot{\theta}\cos\theta + \frac{1}{2}ml^2\dot{\theta}^2 + mgl\cos\theta \end{aligned} \tag{2.37}$$

From now on, we will simplify notation writing x instead of x_c

Euler-Lagrange Equations:

General form:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = F_i \tag{2.38}$$

For our system:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} = F \tag{2.39}$$

and

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0 \tag{2.40}$$

Where:

$$\frac{\partial L}{\partial x} = 0$$

Developing the first equation 2.39:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) = \frac{d}{dt}((M+m)\dot{x} + ml\dot{\theta}\cos\theta) = (M+m)\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta$$

Thus, the first equation becomes:

$$(M+m)\ddot{x} + ml(\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta) = F \tag{2.41}$$

Addressing the second equation 2.40:

$$\frac{\partial L}{\partial \theta} = -mgl\sin\theta$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = \frac{d}{dt}(ml\dot{x}\cos\theta + ml^2\dot{\theta}) = m\ddot{x}l\cos\theta - ml\dot{\theta}\dot{x}\sin\theta + ml^2\ddot{\theta}$$

The second equation becomes:

$$ml\ddot{x}\cos\theta - ml\dot{\theta}\dot{x}\sin\theta + ml^2\ddot{\theta} + mgl\sin\theta = 0$$

Now, adding damping proportional to the angular velocity $\dot{\theta}$, and with damping coefficient c, the damping torque is $-c\dot{\theta}$. We calculate the value of c using the relationship for a damped harmonic oscillator for a pendulum and including the damping torque in the second equation, we get:

$$c = 2\zeta \cdot m\sqrt{g \cdot l} = 0.57\frac{\mathrm{N}\cdot\mathrm{m}\cdot\mathrm{s}}{\mathrm{rad}} \tag{2.42}$$

$$ml\ddot{x}\cos\theta - ml\dot{\theta}\dot{x}\sin\theta + ml^2\ddot{\theta} + c\dot{\theta} + mgl\sin\theta = 0 \tag{2.43}$$

Non linear Equations of Motion :

$$(M+m)\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F \tag{2.44}$$

$$ml^2\ddot{\theta} + ml\ddot{x}\cos\theta + c\dot{\theta} - ml\dot{\theta}\dot{x}\sin\theta + mgl\sin\theta = 0 \tag{2.45}$$

Linearizing the equations around equilibrium ($\theta = 0$):

- Assuming $\sin\theta \approx \theta$ and $\cos\theta \approx 1$ for small θ angles,
- Neglecting the terms with $\dot{\theta}^2$ and with $\dot{x}\dot{\theta}$

The equations become:

Linearized Equations of Motion :

$$(M+m)\ddot{x} + ml\ddot{\theta} = F \tag{2.46}$$

$$ml^2\ddot{\theta} + ml\ddot{x} + c\dot{\theta} + mgl\theta = 0 \tag{2.47}$$

State-space Representation:

$$\text{State vector } \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}$$

Isolating $\ddot{x}$ and $\ddot{\theta}$ from equation 2.47

$$\ddot{\theta} = -\frac{1}{l}\ddot{x} - \frac{c}{ml^2}\dot{\theta} - \frac{g}{l}\theta \tag{2.48}$$

$$\ddot{x} = -l\ddot{\theta} - \frac{c}{ml}\dot{\theta} - g\theta \tag{2.49}$$

Substituting 2.48 in 2.46:

$$(M+m)\ddot{x} + ml(-\frac{g}{l}\theta - \frac{1}{l}\ddot{x} - \frac{c}{ml^2}\dot{\theta} = F$$

$$M\ddot{x} - mg\theta - \frac{c}{l}\dot{\theta} = F$$

$$\ddot{x} = \frac{mg}{M}\theta + \frac{c}{Ml}\dot{\theta} + \frac{1}{M}F \tag{2.50}$$

Substituting 2.48 in 2.46:

$$M(-l\ddot{\theta} - \frac{c}{ml}\dot{\theta} - g\theta) + m(-l\ddot{\theta} - \frac{c}{ml}\dot{\theta} - g\theta) + ml\ddot{\theta} = F$$

$$Ml\ddot{\theta} = -Mg\theta - \frac{Mc}{ml}\dot{\theta} - mg\theta - \frac{c}{l}\dot{\theta} - F$$

$$\ddot{\theta} = -\frac{(M+m)g}{Ml}\theta - \dot{\theta}(\frac{c}{Ml^2} + \frac{c}{ml^2}) - \frac{1}{Ml}F$$

$$\ddot{\theta} = -\frac{(M+m)g}{Ml}\theta - \dot{\theta}c\frac{(M+m)}{Mml^2} - \frac{1}{Ml}F \tag{2.51}$$

$$x_2 = \dot{x}_1 \tag{2.52}$$

$$x_4 = \dot{x}_3 \tag{2.53}$$

Writing equations 2.50, 2.51, 2.52 and 2.53 in matrix form:

State-space equations:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{mg}{M} & \frac{c}{Ml} \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{(M+m)g}{Ml} & -c\frac{(M+m)}{Mml^2} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ \frac{1}{Ml} \end{bmatrix} F \tag{2.54}$$

I won't demonstrate it, but the system results:

- Stable (not asymptotically since two eigenvalues have null real part)
- Controllable (rank of the controllability matrix equals 4, i.e. the number of states)
- Fully observable (rank of the observability matrix is 4)

Having checked this, we can apply a controller to the system.

2.2.2 Structure of the MPC controller for real-time tracking with water spilling avoidance

Our objective will be to do an MPC for real-time trajectory tracking, where we have an input reference signal at each time step, which is a measurement of the reference trajectory at that time step. Since we are working in a simulation environment, we define the reference trajectory a priori and take at each time step a component of the trajectory to simulate a real-time measurement.

The reference trajectory that the system will have to imitate in real time consists of starting in a position at rest and stopping in a final position, while avoiding spilling water. This trajectory represents the most probable way the user would move the glass. Assuming a linear superposition of orthogonal spatial coordinate, this can be formulated on the 2-D model in moving the cart from position x_0 and to stop in position x_f without letting the angle θ of the pendulum reach θ_{max}.

Structure of Tracking Model Predictive Control of a Linear Time Invariant system

The system is a SIMO (single input, multi output) since we control two outputs being the position of the cart x and the angle of the pendulum θ with one input being the force on the cart.

Prediction Model:

$$\mathbf{x}(k+1) = A\mathbf{x}(k) + Bu(k) \tag{2.55}$$

$$y(k) = C\mathbf{x}(k) + Du(k) \tag{2.56}$$

where $\mathbf{x}(k)$, $u(k)$, and $\mathbf{y}(k)$ represent the states, input, and outputs at time step k, and A, B, C, and D are the system matrices.

Cost Function for Real-Time Trajectory Tracking:

$$\begin{aligned} J = \sum_{i=0}^{N-1} \; & [(\mathbf{y}(k+i) - \mathbf{r}_{\mathrm{m}}(k+i))^T Q(\mathbf{y}(k+i) - \mathbf{r}_{\mathrm{m}}(k+i)) + \dots \\ & \cdots + u(k+i)^2 R + \Delta u(k+i)^2 S \end{aligned} \tag{2.57}$$

where:

- $\mathbf{r}_{\mathrm{m}}(k+i)$ is the measured and estimated time-varying reference at each time step k.
- $\Delta u(k+i)$ is the change in the control input between two steps.
- Q is the weight on the tracking error between the predicted output $\mathbf{y}(k+i)$ and the measured trajectory $\mathbf{r}_{\mathrm{m}}(k+i)$.
  Q is a positive semi-definite matrix.
- R is the weight on the control effort.
  R is normally a positive definite matrix, but being the system single input, R is a positive scalar.
- S is a weight matrix penalizing the rate of change of the control inputs.
  Again, being the system single input, S is a scalar.
- N is the prediction horizon.
- M is the control horizon.

Optimization Problem:

$$\min_{u(k),\ldots,u(k+M-1)} J \tag{2.58}$$

Subject to:

$$\mathbf{x}(k+1) = A\mathbf{x}(k) + Bu(k) \tag{2.59}$$
$$\mathbf{x}_{\min} \leq \mathbf{x}(k) \leq \mathbf{x}_{\max} \tag{2.60}$$
$$u_{\min} \leq u(k) \leq u_{\max} \tag{2.61}$$
$$\Delta u_{\min} \leq \Delta u(k) \leq \Delta u_{\max} \tag{2.62}$$

MPC for our specific problem

- State vector:

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} \tag{2.63}$$

- System matrices:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{mg}{M} & \frac{c}{Ml} \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{(M+m)g}{Ml} & -c\frac{(M+m)}{Mml^2} \end{bmatrix} \tag{2.64}$$

$$B = \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ \frac{1}{Ml} \end{bmatrix} \tag{2.65}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.66}$$

$$D = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.67}$$

- Initial conditions of the state vector:

$$\mathbf{x}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.68}$$

- The output variables $y(k)$:

$$y(k) = \begin{bmatrix} x(k) \\ \theta(k) \end{bmatrix} \tag{2.69}$$

- Reference at every iteration
  At every iteration, the reference is the measured signal of the reference position for the cart at step k (z_k) and the reference 0 for angle θ:

$$x_{ref(k)} = z_k \tag{2.70}$$
$$\theta_{ref(k)} = \theta_{ref} = 0 \tag{2.71}$$

- Constraints on the control input:
  These are normally dictated by the actuators, but we suppose that there are no constraints on the control input

- Constraints on the output variables:
  The pendulum's angle has to be within a certain range to avoid overflowing from the glass. The value of $\|\theta_{max}\|$ is given in equation 2.29.
$$-\|\theta_{max}\| < \theta < \|\theta_{max}\| \tag{2.72}$$

- Constraints on the states:
  For safety reasons, we limit the speed of the cart at

$$\|\dot{x}_{max}\| = 0.5 \text{ m/s}, \tag{2.73}$$

  which a reasonable velocity limit of the glass for such an application of direct imitation of the user.

$$-\|\dot{x}_{max}\| < \dot{x} < \|\dot{x}_{max}\| \tag{2.74}$$

Reference trajectory of the cart

As I said before, the cart starts with null speed in $x_0 = 0$ and stops in x_f. The reference trajectory of the user with $x_f = 2$m is shown in Fig. 2.6. The cart should reach the final position in approximately 2.5 seconds.

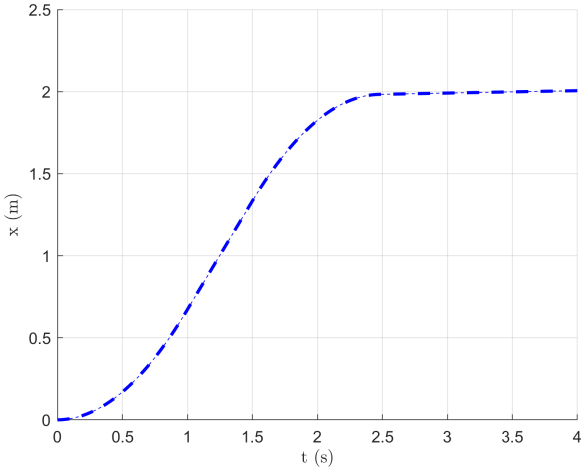


Figure 2.6: Reference trajectory of the cart. The MPC controller will take a sample of this trajectory at every time step to simulate real-time measurements

However, for the tuning of the MPC parameters, and to test the responsiveness and stability of the controller, we will start by first considering a step input as the cart reference trajectory.

2.2.3 Performance parameters for the interpretation of the results

1. Maximum instantaneous tracking error (Ei_{max}):
   The maximum difference between the current cart position and the real-time reference signal, measured instantaneously

2. Cumulative tracking error(E_{cum}):
   The maximum difference between the current cart position and the real-time reference signal, measured cumulatively and squared. More specifically, I chose to calculate it as the sum of squared errors.

3. Maximum overshoot (O_{max}):
   The maximum deviation of the cart position beyond the desired reference position during the response

4. Maximum pendulum angle reached ($\theta_{max(MPC)}$):
   The maximum angle of the pendulum with vertical during the simulation. It is a metric for the stability of the system.

5. Rise time ($R_{t5\%}$):
   It measures the responsiveness of the system to a step reference input on the position of the cart and is defined as the time it takes for the

system response to go from 5 to 95 percent of its final value following a step change in input

2.2.4 Considerations for the Tuning of MPC parameters

The MPC parameters have to be tuned to achieve a good performance of the system. I conducted the tuning experimentally.

1. Prediction and control horizon:
   The shorter the prediction horizon (N), the more responsive the system is and the less the computational cost at every iteration. On the other hand, the higher the prediction horizon, the more accurate the long-term behavior and stability of the system, but the more the computational cost.
   The control horizon (M) is equal or smaller than the prediction horizon. For a chosen value of N, I will experiment different values of M to find the better solution.

2. Weight on the control effort R:
   This weight controls how much the controller penalizes the use of the control input. It is a scalar. A higher weight reduces the magnitude of the control input, leading to smoother, more energy-efficient control but possibly at the cost of less accurate tracking. We suppose that minimizing control effort is not a priority in our control problem, thus we choose a null value:
   $$R = 0 \tag{2.75}$$

3. Weight on the control action variation S:
   This weight controls how much the controller penalizes high variations in the control input. A higher weight ensures a smooth variation in the control action. Again, we suppose that it is not a priority and set it to a null value:
   $$S = 0 \tag{2.76}$$

4. Weight on the tracking error Q
   This weight determines how much importance is placed on minimizing the tracking error. It is a matrix:
   $$Q = \begin{bmatrix} Q_x & 0 \\ 0 & Q_\theta \end{bmatrix} \tag{2.77}$$
   A higher weight will make the controller prioritize accurate tracking over minimizing the control effort. We want to have a responsive system in terms of following the cart trajectory and at the same time keep θ_{max} close to its reference value. But at the same time, we have to consider a limit in the cart velocity as stated in equation 2.73. We test the system with different increasing values of the matrix Q elements.

2.2.5 Results of the simulations

1) Step reference input

- Prediction horizon N:
  For applications where high responsiveness is requested like simultaneous imitation of the user, low values of the prediction horizon are preferred. Thus, we choose a very low value, but not too low, to have also less oscillations(the value is the number of time steps)

$$N = 5 \tag{2.78}$$

- Control horizon M:
  We choose the minimum (the value is the number of time steps):

$$M = 1 \tag{2.79}$$

- Weight on the tracking error Q:
  The higher Q_x, the higher the responsiveness and the lower tracking error in the position of the cart.
  The higher Q_x, the lower the value of the $\theta_{max(MPC)}$, but also the lower the responsiveness.
  The best combination that I obtained is:

$$Q = \begin{bmatrix} 20 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.80}$$

The results of the simulation with the tuned values of the parameters, which are a good compromise in terms of cart position and pendulum oscillations are shown in Fig. 2.7. We observe a smooth and fast trajectory of the cart and oscillating values of the pendulum angle below the maximum allowed. The cart velocity stays below the limit of 0.5 m/s. The performance metrics that we obtain are:

- Ei_{max} doesn't make sense to be measured, since with a step reference it will be 1.
- $$E_{cum} = 143.42 \text{ meters}^2 \tag{2.81}$$
- $$O_{max} = 3.7 \text{ millimeters} \tag{2.82}$$
- $$\theta_{max(MPC)} = 0.406 \text{ radians} < \theta_{max} = 0.5191 \text{ radians} \tag{2.83}$$

- $$R_{t5\%} = 1.87 \text{ seconds} \tag{2.84}$$

We can see a high value of the squared cumulated error, that is due to the overcome of the velocity constraint by the user. We can observe a very low overshoot and good value of rise time, which is strictly related again to the velocity constraint on the cart. The results are satisfying.

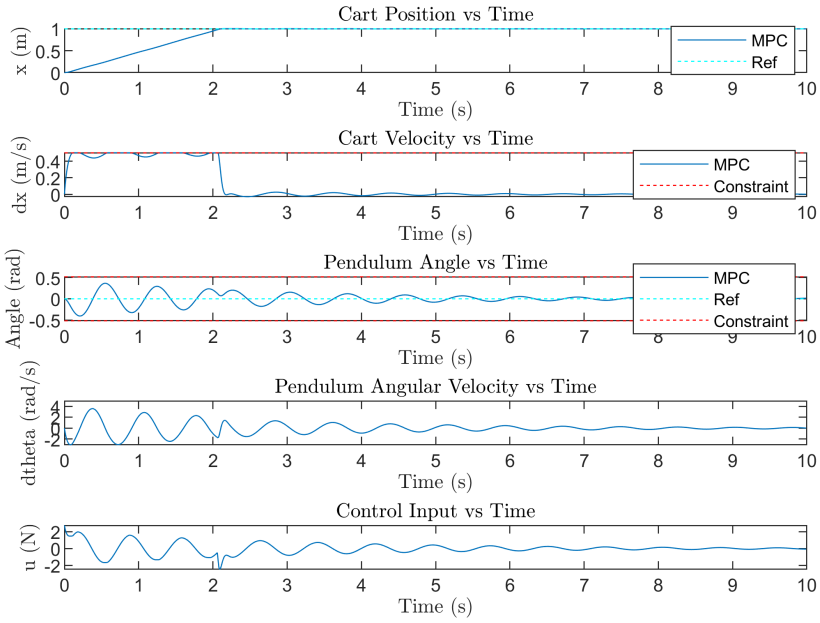


Figure 2.7: State and control action with respect to time in MPC with a step reference input and values $N = 5$, $M = 1$, $Q_x = 20$, $Q_\theta = 1$, $R = 0$ and $S = 0$

2) Real application reference trajectory that overcomes the velocity constraint on the cart

Now we will consider a possible trajectory in a real application, where the user wants to smoothly transport the glass of water from position $x_0 = 0$ to $x_f = 2$ m (Fig. 2.6). More specifically, we study the case where the user reference trajectory for the cart overcomes the cart velocity constraint of 0.5 m/s. The results of the simulation with the already tuned values of the parameters are shown in Fig. 2.8 and Fig. 2.9 and the values of the metrics for the performance are:

- $$Ei_{max} = 0.95 \text{ meters} \tag{2.85}$$

- $$E_{cum} = 140.56 \text{ meters}^2 \tag{2.86}$$
- $$O_{max} = 6.15 \text{ millimeters} \tag{2.87}$$
- $$\theta_{max(MPC)} = 0.3545 \text{ radians} < \theta_{max} = 0.5191 \text{ radians} \tag{2.88}$$

The results are good, but the tracking error is high due to the contraint on the cart velocity, as predicted. We will compare these data with a case in which the user doesn't overcome the constraint.

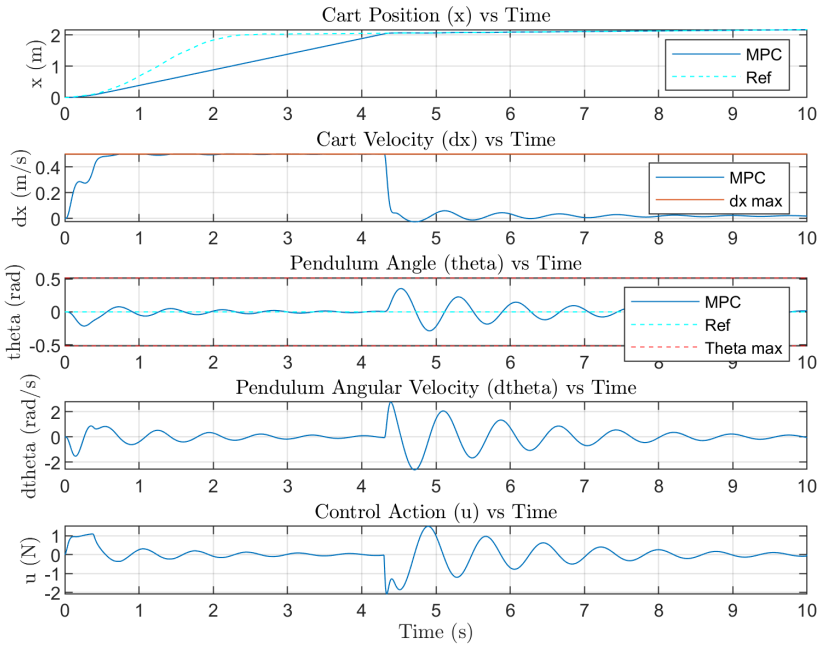


Figure 2.8: State and control action with respect to time in MPC simultaneous tracking with values $N = 5$, $M = 1$, $Q_x = 20$, $Q_\theta = 1$, $R = 0$ and $S = 0$.

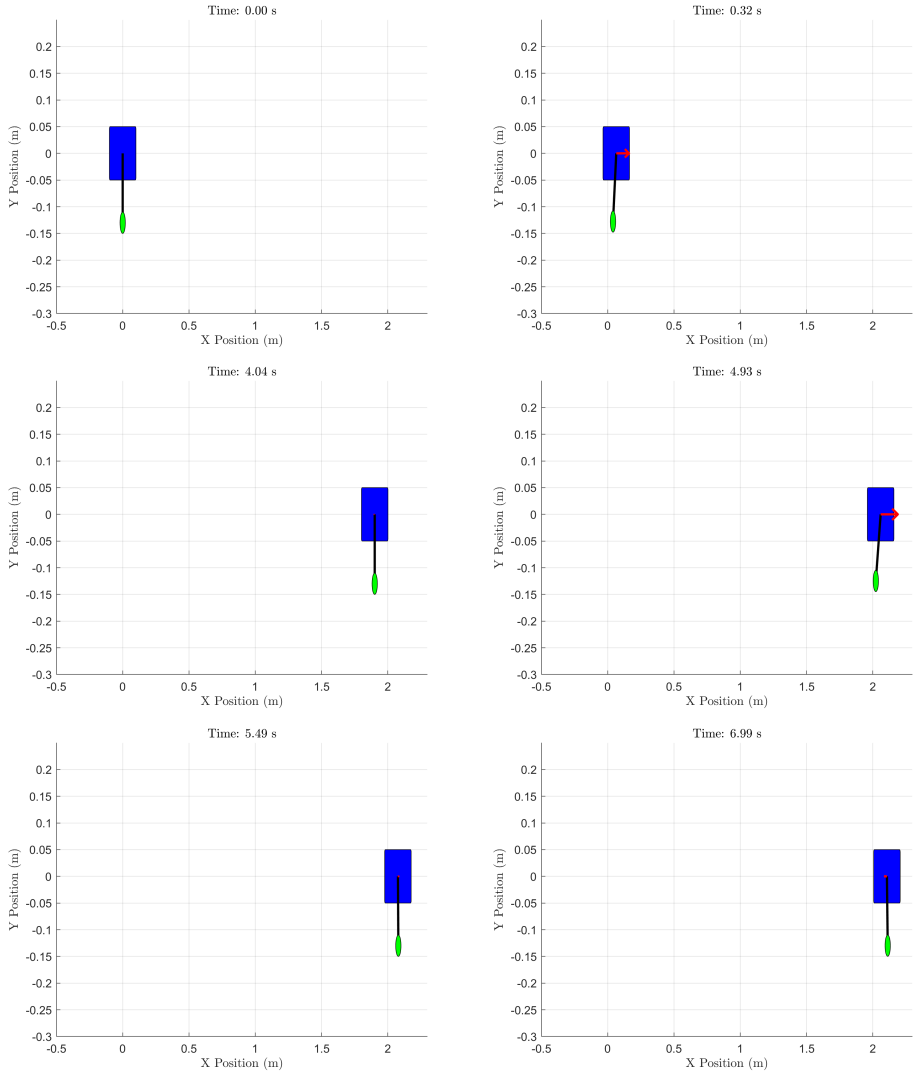


Figure 2.9: Sequence of snapshots of MPC tracking with values $N = 5$, $M = 1$, $Q_x = 20$, $Q_\theta = 1$, $R = 0$ and $S = 0$. The red arrow represents the direction and magnitude of the control action at that instant.

3) Reference trajectory that doesn't overcome the velocity constraint on the cart

In this third scenario, the user reference trajectory for the cart is slower and doesn't overcome the speed limit of the cart of 0.5 m/s. The reference trajectory is shown in Fig. 2.10.

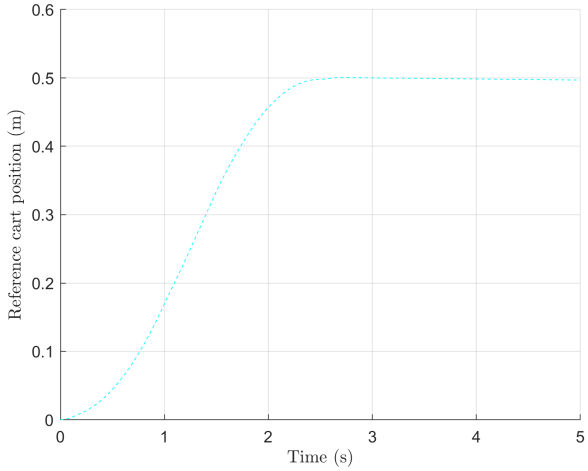


Figure 2.10: Reference trajectory of the cart without overcoming of the velocity constraint on the cart. The MPC controller will take a sample of this trajectory at every time step to simulate real-time measurements

With the same tuned parameters, the states and input versus time are shown in Fig. 2.11. This time we observe a smooth trajectory overlapping the reference signal, extremely small pendulum angles and nearly absent oscillations. The performance metrics are summed up in Table 2.2.

	$\|\dot{x}\| > \|\dot{x}_{max}\|$	$\|\dot{x}\| < \|\dot{x}_{max}\|$
$\text{Ei}_{max}[m]$	0.95	**0.0139**
$\text{E}_{cum}[m]$	140.56	**0.0212**
Omax [mm]	6.15	**0.55**
$\theta_{max(MPC)}[rad]$	0.3545	**0.0358**

Table 2.2: Performance metrics of MPC simultaneous reference tracking for the case in which the user overcomes the velocity constraint of the cart and the other case in which the constraint is not overcome.

We can see that if $\|\dot{x}\| < \|\dot{x}_{max}\|$ we obtain very good results and the controller succeeds in copying almost perfectly the user.

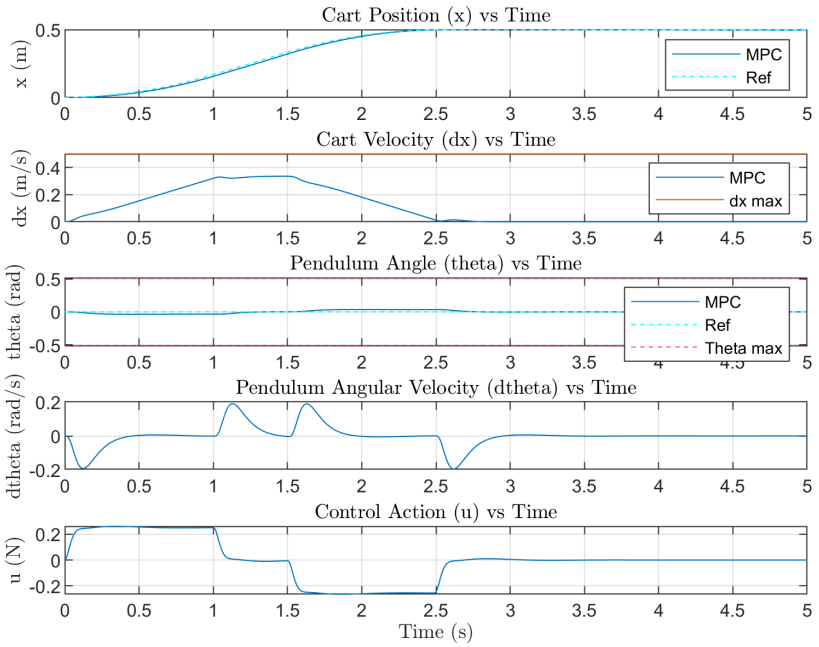


Figure 2.11: State and control action with respect to time in MPC simultaneous tracking with values $N = 5$, $M = 1$, $Q_x = 20$, $Q_\theta = 1$, $R = 0$ and $S = 0$.

Conclusion and future work

This thesis has explored the implementation of robotic assistance tasks for humans using a variety of control methods and simulation environments. We began by developing a potential field-based motion planning algorithm, which proved effective for obstacle avoidance in assistive tasks. This approach was successfully applied to two key scenarios: robotic assistive feeding and dressing. In the feeding task, we demonstrated the algorithm's ability to navigate around static and dynamic obstacles, such as the user's hands, while maintaining a safe and efficient trajectory to the user's mouth. For the dressing task, we adapted the algorithm to guide a simulated sleeve along a user's arm, accounting for potential arm movements.

We then progressed to more complex scenarios involving active user control and collaborative object manipulation. This included a collaborative box transport task, where we implemented direct imitation control with a Kalman filter to handle potential motor impairments modeled as noise in the user's hand movements. Finally, we tackled a challenging glass manipulation task using Model Predictive Control (MPC) for real-time trajectory tracking. This task incorporated the dynamics of water sloshing in the glass, demonstrating our ability to handle more complex system dynamics while meeting multiple control objectives.While the results of the simulations are promising, there are several important considerations and steps to be taken for future development towards real-world applications.

The simulation environment provided a valuable platform for testing and refining our algorithms. However, it's important to acknowledge that real-world implementation will introduce additional complexities. One of the primary challenges will be incorporating full system dynamics. Although I implemented the dynamics of a model of a glass filled with water, it is as well required to derive and implement the full equations of motion for the manipulator and any objects it interacts with in all the applications.

Another crucial aspect for future development is the integration of realistic sensor models and robust sensor fusion algorithms. Implementing and testing with realistic sensor models, including RGB-D cameras, force/torque sensors, and proprioceptive sensors, will be essential for ensuring the robustness of our algorithms in noisy environments.

The potential field method we developed for obstacle avoidance proved

effective in our simulations. However, to handle more complex and dynamic real-world environments, it may be beneficial to integrate this approach with other planning methods like Rapidly-exploring Random Trees (RRT) or probabilistic roadmaps for global path planning. This hybrid approach could enhance the system's ability to navigate through cluttered spaces while maintaining the reactive capabilities of the potential field method.

Moreover, for a physical implementation, developing detailed models of the manipulator's actuators will be crucial. This includes considering torque limits, response characteristics, and control algorithms. Designing low-level control systems that can accurately execute the desired motions while respecting physical constraints will be a key challenge.

Safety is essential in assistive robotics, especially given the close proximity between the robot and the user. Future work should focus on developing and implementing comprehensive safety systems, including collision detection, force limiting, and emergency stop procedures. These systems need to be rigorously tested and validated to ensure user safety under all operating conditions.

For each specific assistive task (feeding, dressing, etc.), more detailed models and algorithms should be developed to account for task-specific challenges. For example, in feeding tasks, we need to consider food manipulation and the variability in users' eating behaviors. Implementing adaptive control strategies that can adjust the robot's behavior based on the user's capabilities and preferences will be key to creating personalized and effective assistive systems.

Furthermore, as we progress towards real-world implementation, developing physical prototypes will be an essential step. This will involve selecting appropriate hardware, integrating sensors and actuators, and implementing our control algorithms on real-time embedded systems. Through this process, we can identify and address practical challenges that may not have been apparent in simulation.

By addressing these areas, future work can close the gap between simulation and real-world implementation, bringing us closer to practical and effective robotic assistance systems that can significantly improve the quality of life for individuals requiring assistance in daily activities.

List of Figures

List of Tables

Bibliography

[1] E.G. Christoforou, S. Avgousti, and N. Ramdani. The upcoming role for nursing and assistive robotics: Opportunities and challenges ahead. *Frontiers in Digital Health*, 2, 2020.

[2] Naonori Kodate, Hasheem Mannan, and Sarah Donnelly. Care robots as enabling assistive technology: implications for quality of life and disability policy. In *The Routledge International Handbook of Disability and Rehabilitation*, pages 60–74. Edward Elgar Publishing, 2023.

[3] I. Papadopoulos, C. Koulouglioti, R. Lazzarino, and S. Ali. Enablers and barriers to the implementation of socially assistive humanoid robots in health and social care: a systematic review. *BMJ Open*, 10(1), 2020.

[4] Steven W Brose, Douglas J Weber, Ben A Salatin, Garret G Grindle, Hongwu Wang, Juan J Vazquez, and Rory A Cooper. The role of assistive robotics in the lives of persons with disability. *American Journal of Physical Medicine & Rehabilitation*, 89(6):509–521, 2010.

[5] B Radder, GB Prange-Lasonder, AIR Kottink, JH Stienen, and HT Hermens. Home rehabilitation supported by a wearable soft-robotic device for improving hand function in older adults: A pilot randomized controlled trial. *PloS one*, 14(8), 2019.

[6] Alireza Mohebbi. Human-robot interaction in rehabilitation and assistance: a review. *Current Robotics Reports*, 1(2):61–71, 2020.

[7] Maria Manti, Alessandro Pratesi, Enrico Falotico, Paolo Dario, and Cecilia Laschi. Soft assistive robot for personal care of elderly people. In *IEEE International Conference on Biomedical Robotics and Biomechatronics*, pages 833–838. IEEE, 2016.

[8] Z. Chen, L. Ma, and Z. Shao. Path planning for obstacle avoidance of manipulators based on improved artificial potential field. In *2019 Chinese Automation Congress (CAC)*, pages 1148–1153. IEEE, 2019.

[9] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *1986 IEEE International Conference on Robotics and Automation. Proceedings*, volume 2, pages 500–505. IEEE, 1986.

[10] B. N. Biswas, S. Chatterjee, S. P. Mukherjee, and S. Pal. A discussion on euler method: A review. *Electronic Journal of Mathematical Analysis and Applications*, 1(2):294–317, July 2013.

[11] Jihong Zhu et al. Do you need a hand? – a bimanual robotic dressing assistance scheme. *arXiv preprint arXiv:2301.02749*, 2023.

[12] Wikipedia. Rodrigues' rotation formula. `https://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula`.

[13] Hanjun Song Tapomayukh Bhattacharjee, Gilwoo Lee and Siddhartha S. Srinivasa. Towards robotic feeding: Role of haptics in fork-based food manipulation. *IEEE Robotics and Automation Letters*, 2019.

[14] Universal robots ur5/ur5e. `https://www.mybotshop.de/Universal-Robots-UR5-UR5e_1`.

[15] Intel realsense depth camera d435i. `https://www.mybotshop.de/Intel-RealSense-Depth-Camera-D435i_1`.

[16] Laser tracker api 3. `https://micro3d.it/en/prodotto/laser-tracker-api-3/`.

[17] End of arm accessories - force torque sensors. `https://unchainedrobotics.de/en/category/end-of-arm-effectors/end-of-arm-accessories/force-torque-sensors`.

[18] John Kirtley. Kalman filter lecture mit notez. `https://web.mit.edu/kirtley/kirtley/binlustuff/literature/control/Kalman%20filter.pdf`.

[19] Luca Guagliumi, Alessandro Berti, Eros Monti, and Marco Carricato. A simple model-based method for sloshing estimation in liquid transfer in automatic machines. *IEEE Access*, 9, September 2021. Received June 10, 2021; accepted September 15, 2021; published September 20, 2021.

[20] H Melkas L Sørensen, DT Johannesen. Care-receivers with physical disabilities' perceptions on having humanoid assistive robots as assistants: a qualitative study. *BMC Health Services Research*, 24(523), 2024.

[21] Y. Kim D. Gallenberger, T. Bhattacharjee and S.S. Srinivasa. Transfer depends on acquisition: Analyzing manipulation strategies for robotic feeding. *ACM/IEEE International Conference on Human-Robot Interaction.*, 2019.

[22] Jorge Pomares and Fernando Torres. Time independent tracking using 2-D movement flow-based visual servoing. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2541–2546. IEEE, 2005.

[23] Mohammed Alhajeri and Masoud Soroush. Tuning guidelines for model-predictive control. *I&EC Research*, 4(10):1–30, October 2021. Fourth Revised Version.