MASTER

Towards Robot Tossing by Demonstration through Intuitive Teleoperation

van Gorp, Teun I.C.

Award date:
2022

Department of Mechanical Engineering

Dynamics and Control Section
Manufacturing Systems Engineering

Towards Robot Tossing by Demonstration through Intuitive Teleoperation

Master of Science Thesis

T.I.C. van Gorp (1386573)
DC 2022.038

Coaches:	ir. J.J. van Steen	
	ir. M.J. Jongeneel	
	ir. S. de Looijer	(Vanderlande)
	dr. ir. I.A. Kuling	
Supervisor:	dr. ir. A. Saccon	

Eindhoven, April 2022

Abstract

Due to the popularity of e-commerce and the lack of labor power, logistic warehouses could benefit from faster parcel handling. Currently, many tasks are handled by robot manipulators. Throughput of these systems is often limited because picking and placing the objects happens at zero relative velocity of the robotic arm. To increase throughput, tossing is considered a possible solution. This enables robots to handle objects faster, and to put objects outside its kinematic reach.

The required controllers for robotic tossing have to deal with complex challenges related to flight dynamics, acquiring pre-toss conditions, the wide variety of objects, and impact dynamics when the object hits the surface. Because of these reasons, robotic solutions that are capable of tossing objects onto desired target surfaces, is still uncharted territory. In this work, we propose an alternative method to achieve this, while avoiding a great deal of controller complexity.

Our proposed solution to push the development of robotic tossing solutions is the development of a teleoperation system that enables humans to control the robot's motion and gripper. The idea is to enable the operator to perform robotic tossing demonstrations in an intuitive way, through natural body movements for tossing. This way, human intuitivity and sensorimotor skills can be exploited for the development towards robotic tossing trajectories. The realized teleoperation system's main contributions are its ability to quickly explore various robotic tossing methods, to see how objects can effectively be tossed. This contributes to our understanding of which ways of robot tossing are effective, and to identify the limitations of robot tossing.

For industrial applications, it is essential that the outcome of a robot toss is predictable, and can be performed reliably. Besides the development of the teleoperation system, we also explored the industrial applicability of robot tossing. In an experimental setup, we analyzed if repetitions of tossing trajectories can lead to similar tossing outcomes. We showed that the outcome of a tossing trajectory can be repeated with sufficient precision for application in an industrial setting. We find that for tossing with relatively low velocities, where the angle between the bottom surface of the box and the conveyor is relatively small, the outcome of a toss can be repeated quite reliably. When tossing at higher velocities, where the box makes impact with a large angle between its bottom surface and the conveyor, the outcome of a toss is less predictable.

Acknowledgements

I wrote this thesis to conclude the graduation period of the Master of Science program in Dynamics and Control in Mechanical Engineering, at Eindhoven University of Technology. The presented work was executed as a collaboration between Vanderlande and TU/e. During this great learning experience I had the pleasure to work with great colleagues, without them, leading to the exchange of ideas and discussions, I would not have been able to deliver this work. Although I wrote my thesis on my own, I find it inappropriate to write in first-person singular. As we progress through the argument, I will speak in the first-person plural.

First, I want to acknowledge the people that gave me the opportunity to work on this project. I would like to thank my TU/e supervisor, dr. ir. Alessandro Saccon and coach dr. ir. Irene Kuling, for guiding me through the process of graduating, and supporting my research through fruitful discussions. Your critical feedback, honest opinions and enthusiasm have helped me to improve the quality of my work, and to learn a lot. I would also like to thank my coach from Vanderlande ir. Stijn de Looijer. I also greatly appreciate your flexibility and time for helping, especially in emphasizing the industry's point of view in discussions.

At the start of this project, I had few experience on topics related to software programming and QP control, this made the implementation of a commercial grade VR device to control the motion of an industrial grade robot manipulator was very challenging. Therefore, I would like to express special thanks to ir. Jari van Steen, who is one of my coaches during this project. Your patience, suggestions, and feedback helped me a great deal in achieving my project goals, and in understanding the software framework, which allowed me to bring my theoretical knowledge to a real-life application. I am also very grateful to my other coach, ir. Maarten Jongeneel, your vision helped me get more insight in the subject of this project, and your clear ways of explaining and honest opinions helped me in a lot of discussions.

Besides my supervisors and coaches, I also want to thank Vanderlande for allowing me to use their facilities, and providing the robot setup. Thanks to dr. ir. Femke van Beek for allowing me to use the Vive VR Pro kit at the Haptics lab while the hardware for my project was being delivered late. I owe gratitude to ir. Sander Dingemans, who helped me a lot with performing experiments. To all others that I met during this project, enjoyed the many times working in the lab, which made it easier and more fun to work through the COVID-19 period.

I also want to thank my friends and family, for their support and welcome distraction from this graduation project. Special thanks to my parents, sisters, brother, and girlfriend for their support, encouragement and love. Without you, I could not have completed and enjoyed my studies in the way that I did.

I hope you enjoy reading this thesis.

Teun van Gorp
Eindhoven, April 2022

Contents

Nomenclature

Roman symbols

A, B, ...	coordinate frames
$[A]$	orientation frame associated to A
$A[B]$	frame with origin $\mathbf{o}_A$ and orientation $[B]$
B	box rigid body frame
c	cost function
C	conveyor rigid body frame
$\mathbf{C}$	vector of Coriolis and centrifugal forces
$\mathbf{d}_{\mathbf{a},\mathbf{b}}$	distance between points **a** and **b**
$\mathbf{D}$	6×6 inertia matrix
e	error
$\mathbf{E}$	task error function
$\mathbf{g}$	vector of gravity force
${}^{A}\mathbf{H}_{B}$	4×4 homogeneous transformation matrix from B to A
${}^{A}\mathbf{J}_{B}$	Jacobian describing the velocity of B w.r.t. A
k_d	scalar derivative stiffness gain
k_p	scalar proportional stiffness gain
$\mathbf{o}_A$	origin of A
$\mathbf{p}$	arbitrary point
${}^{A}\mathbf{p}$	coordinates of **p**, expressed in A
$\mathbf{Q}$	6×6 symmetric Hessian matrix
$\mathbf{q}$	joint state
${}^{A}\mathbf{R}_{B}$	rotation matrix from B to A
S	robot's end-effector frame
t_o	OptiTrack measurement time
$\mathbf{u}$	vector of motor torques
U	UR10 base frame
V	leader device frame
w	task weight
x	vector of optimization variables $\in \mathbb{R}^n$

Greek symbols

α	reference position scaling
$\Delta\mathbf{p}$	difference between minimum and maximum value in **p**
Δt	controller time step
δ_i	influence distance between two points for a constraint to act
δ_s	safety distance between two points that will never be reached
δt	a constant time step
θ	angle
ξ	velocity damping coefficient
σ	standard deviation
Σ	3×3 covariance matrix

ω	angular velocity scalar
$\boldsymbol{\omega}$	angular velocity vector

Subscripts and Superscripts

$\vec{(\cdot)}$	geometric vector
$\dot{(\cdot)}$	first time derivative
$\ddot{(\cdot)}$	second time derivative
$\\|(\cdot)\\|$	euclidean norm
$\bar{(\cdot)}$	mean
$\overline{(\cdot)}$	upper limit
$\underline{(\cdot)}$	lower limit
$(\cdot)^T$	transpose
$(\cdot)^{-1}$	inverse
$(\cdot)^{\text{ref}}$	reference quantity
$(\cdot)^d$	desired quantity
$(\cdot)^{\max}$	min quantity in set of values
$(\cdot)^{\min}$	max quantity in set of values
$(\cdot)^{\wedge}$	'Hat' operator: vector to matrix representation of algebra element
$(\cdot)^{\vee}$	'Vee' operator: matrix to vector representation of algebra element
$(\cdot)^{\text{pos}}$	quantity related to the position task
$(\cdot)^{\text{ori}}$	quantity related to the orientation task
$(\cdot)_t$	quantity at time t
$\log(\cdot)$	logarithm

Special Operators, Sets, and Notations

SO(3)	the Special Orthogonal group in 3 dimensions
$\mathfrak{so}(3)$	the Lie algebra of SO(3)
k_{pressed}	button k pressed
k_{released}	button k released
$\mathbf{b}^f$	box' rest pose in 2D
$\mathbf{c}$	conveyor pose in 2D
$\mathbf{p}^b$	box pick pose in 2D

Abbreviations

DOF	Degree of Freedom
API	Application Programming Interface
ROS	Robot Operating System
CNRS	Centre National de la Recherche Scientifique
HMD	Head Mounted Display
AIST	National Institute of Advanced Industrial Science and Technology
JRL	Joint Robotics Laboratory
SDK	Software Development Kit
QP	Quadratic Programming
VR	Virtual Reality
I/O	Input/Output
FSM	Finite-State Machine
GJK	Gilbert, Johnson and Keerthi

Chapter 1

Introduction

1.1 Context

The popularity of e-commerce has lead to significant growth of the worldwide parcel industry over the past decade, and the recent consequences of the COVID-19 pandemic pushed this growth even further in the last two years. Instead of spending time to go out shopping at physical stores, people are buying clothing, tech products, books, daily groceries, and more products online. The parcel delivery industry faces a major challenge for increasing efficiency in both speed and costs in order to meet the demand. Due to labor scarcity, and the need for increased productivity and flexibility at the lowest possible costs, automation is considered a key enabler to face this challenge [1]. By investing in the development of smarter, faster and more flexible robotic solutions, the consequences of unfavorable demographic developments (such as the declining availability, motivation and qualification of logistics personnel) [2], and the associated declining number of employees, can be mitigated.

In order to increase productivity in logistics warehouses, several automation systems have already been implemented. However, in the field of logistics, still 80% of the tasks worldwide remain being operated manually [1]. Humans possess well-developed sensory and motor skills that allow to effectively handle a wide variety of objects, in large quantities, various combinations, and limited space. As a result, major sorting hubs still rely on manpower, and employees spend their time packing, unpacking, sorting, realigning, and parcel feeding. Especially feeding parcels into warehouse systems often remains based on human labor. This process is complex because parcels often arrive on unsorted pallets or containers, and come in a wide variety. Processing these objects at a high speed requires a high versatility (the extent of the variety of parcels that human operators or a system can process). Industrial robots were, until recently, immobile and non-intelligent machines that were able to perform simple repetitive tasks with a high level of speed and accuracy. These abilities are sufficient for simple production processes, but often they are not sufficient for all logistics activities, such as the infeed process. An example of an automated solution for the infeed process is the Smart Item Robotics (SIR) solution from Vanderlande, see Figure 1.1. This system uses vision software to identify and locate objects, and is equipped with a custom vacuum gripper, which enables the system to handle a wide variety of objects.

Since the whole logistics industry faces the challenges mentioned above, a growing number of developments in robotic solutions focus on gaining the required versatility. Various research efforts have tried to address this challenge by, for example, redesigning the robot hardware [3][4], trajectory optimization [5] [6], or increasing the robot's response time through new control strategies [7] [8]. One of the proposed solutions to increase robot versatility is to explore more agile handling techniques. Unlike human operators, current robots are not capable of tossing packages of arbitrary weight and size, to precise locations. This is originates from the fact that robots are not specifically designed to perform tossing operations, and because this significantly increases the

Figure 1.1: Smart Item Robotics solution of Vanderlande [10].

robot control. However, if robots or robot controllers were developed such that they can pick and toss packages, performance in terms of speed can be increased, and might (at some point) even exceed human performance. Besides reaching higher productivity, tossing also enables robots to place items outside its kinematic range. This could even decrease the number of robots required in logistics warehouses [9]. However, using robot manipulators for precise pick-and-toss operations in semi-structured environments is challenging. Especially predicting the motion of a parcel depends on many factors: from pre-toss conditions (e.g. tossing velocity, direction, and pick pose of the object) to the properties of the package (e.g., mass distribution, shape, and size), flight dynamics (e.g. aerodynamics) and its environment (e.g. landing surfaces), some of these parameters are difficult to estimate.

This work continues in the direction of agile handling. Teleoperation or kinesthetic teaching can be used to demonstrate certain tasks to robots. For certain applications, both these methods are user-friendly, and allow operators to intuitively control a robot. However, in kinesthetic teaching, by manually guiding the robotic manipulator in free space, demonstration of tossing operations is not straightforward. It is typically difficult to teach tasks kinesthetically if these require synchronization between multiple of the robot's links. One of the most major drawbacks of kinesthetic learning is that the human must generally use more of their own degrees of freedom (DOF) to move the robot, than the number DOFs they are trying to control. To this end, the proposed solution is to develop an intuitive teleoperation system that allows human operators to perform tossing demonstrations, such that human intuitivity and sensorimotor skills can be exploited for quick generation of tossing trajectories. To the best of the author's knowledge, teleoperation has never been used before to perform tossing demonstrations. While doing so, waypoints or entire trajectories, consisting of joint angles, tool positions and orientations, can be recorded and stored. In the long term, the recorded trajectories could be adapted to overcome discrepancies in the environment and/or objects, and ensure successful tossing. Besides offering an advantage in generating tossing trajectories, through quick demonstration of tossing to predefined target locations, various tossing strategies (motion trajectories) can be assessed. Moreover, by looking at the results of tosses, and repetitions of tosses, the predictability of robot tossing can be studied.

This work is part of the I.AM. project[1], which is an academic-industry collaboration, involving Eindhoven University of Technology and Vanderlande Industries (among other universities and companies), that aims at leveraging robotics technology to strengthen European leadership in this field [11]. I.AM. particularly focuses on the development of solutions for the aforementioned, and contributes to the challenges faced today by the logistics industry. For this research, the facilities at the Vanderlande Innovation lab at the TU/e campus are used. For an introduction of this lab, please refer to Appendix A.

[1]I.AM. is a European Union H2020 project, under GA 871899, see `https://i-am-project.eu/`.

1.2 Related Work

1.2.1 Pick-and-Place Robot Manipulators

Traditional pick-and-place operations of robotic manipulators, that handle a variety of objects, often require accurate information of the objects to be manipulated. For each object, a predetermined picking location is in place and a fixed motion trajectory brings objects to the appropriate places. There are several picking systems available that function by explicitly modeling the contact forces using prior knowledge of the object size, weight, shape, and inertia [12] [13]. Due to the need for warehouse flexibility [14] such that a large and ever-changing variety of items can be handled at high rates, recent research is pointed towards picking and placing of novel objects. Even though knowledge about novel objects can be roughly estimated by vision techniques, precisely estimating how to pick-and-place objects in semi-structured environments remains challenging [15]. However, many advances have been made in the development of pick-and-place operations where manipulators do not require prior knowledge about objects [16] [17] [18]. Systems have been developed that can accurately and robustly recognize target objects in scattered environments [6]. These systems are able to perform a variety of robotic tasks by integrating a set of state-of-the-art techniques in deep learning, computer vision, and trajectory optimization. Moreover, there are picking approaches available, for adaptive picking and trajectory planning, which enables a robot arm to dynamically track and pick objects moving in arbitrary trajectories [19] [20] [21]. Figure 1.2 shows a range of objects from the 2016 edition of Amazon's Picking Challenge to illustrate the wide range of objects encountered in the logistics industry. This edition was won by Team Delft [22], by successfully stowing 11 out of 12 objects in 7 minutes and 10 seconds. One year later, the winning Cartman robot system was able to successfully pick all objects [23], but still employed a very low picking rate of only 120 picks an hour.

The picking solutions of Amazon's Picking Challenge illustrate that the solutions mentioned above are often not able to compete with human versatility in pick-and-place applications. A more versatile pick-and-place solution, which is already implemented in industrial applications, is the Smart Item Robotics (SIR) system (see Figure 1.1), as introduced earlier. Another example is Schäfer's fully automated robot cell *Robo-Pick*, which allegedly picks more than 2,400 items per hour. Compared to human operators, which approximately pick 400 items per hour [25], this is a huge improvement. However, due to the robot's design, it can only pick objects straight from above, limiting its versatility. Looking at existing applications of pick-and-place solutions, there are no robots that reach human versatility and flexibility. The large number of developments in logistic automation, however, offer perspective for new solutions that will gradually reach the human performance level. More recent developments led to even more robust solutions, that can handle a wider variety of objects. Examples of recent developments are the RightPick Platform of RightHand Robotics [26], and GRIPP's Robot Piece Picking solution [27], which allegedly pick up to 1,200 items per hour.

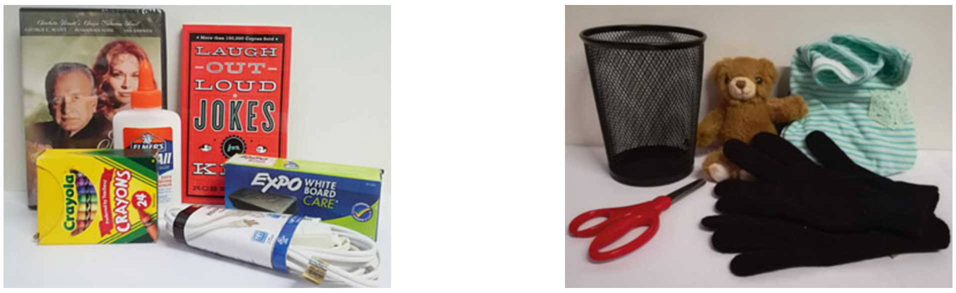

(a) Structured objects.　　(b) Unstructured objects.

Figure 1.2: A handful of the items of Amazon's Picking Challenge in 2016, illustrating the wide range of objects encountered by logistics [24].

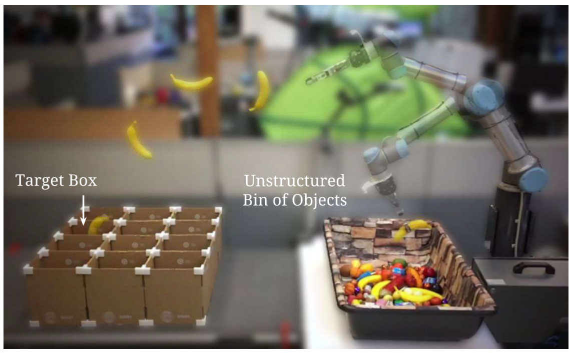


Figure 1.3: TossingBot learned to pick arbitrary objects from an unstructured bin, and to toss them into target boxes located outside its maximum kinematic reach range [29].

1.2.2 Dynamic Object Manipulation

In this research project, we aim to contribute to improving robot performance for pick-and-place tasks, through development of a method that allows a robot to perform agile movements using teleoperation. In literature, numerous papers treat the challenge of improving the agility of robot manipulators. In this work, the definition of [28] is adopted, where agility is defined as "the ability of a robot system to succeed in an environment of continuous and unpredictable change by reacting efficiently and effectively to changing factors". While there is no general agreement in the literature for robot agility, this concept is consistent with its proposed definitions that include not only robot agility, but also the agility of the manufacturing process as a whole.

There are numerous examples of researchers achieving to design or to control robots, enabling them to perform agile robot manipulation. In [30], researchers achieved paddle juggling of a ball by a racket attached to a robot manipulator. Moreover, a 2-DOF robot arm was developed that is capable of hitting objects using a bat [31]. Different robotic tossing approaches have been found in literature: link arms strengthened by springs [32], spring shooting devices, and robot manipulators capable of tossing [33] [34]. This work focuses on the category of (agile) robot manipulators. There already exist robot manipulators that can accurately throw a ball, for example, in a moving basket [33]. An industrial robot manipulator that can successfully throw objects on a target location is described in [34]. However, in these publications, the authors provided the robot with explicit knowledge about the object. A system that can successfully pick arbitrary objects from a bin, and toss them in the desired bin, is Google's TossingBot [29], see Figure 1.3. This robot is trained by a learning algorithm, its success rate of 84.7%, however, is not sufficient for industrial application. In this research project, also box-conveyor impacts must be considered, since a package has to land on a moving conveyor belt, preferably with the barcode up. Tossing in a bin, like the TossingBot does, is much simpler: the ballistic motion and release control are key, but impact dynamics can be ignored. In literature, there is no robot manipulator solution capable of tossing boxes onto target surfaces. Unfortunately, there is no information in literature regarding the actual performance gain through tossing objects. Only in the literature about Google's TossingBot [29], it is reported that pick-and-toss could increase the throughput by almost 10%, compared to traditional pick-and-place operations.

1.2.3 Intuitive Teleoperation for Demonstration

In this work, the development of a teleoperation that allows human operators to control a robot manipulator is presented. By enabling operators to control the motion of a robot manipulator, human intuition can be exploited. This approach has not been used before to teach or gain insights in robotic tossing. However, in literature, telerobotics [35] is a widely explored field of research, including perspectives like: sensor integration, hardware development, control theory, and control architectures. In [36], a teleoperation framework that can quickly map operator movements to

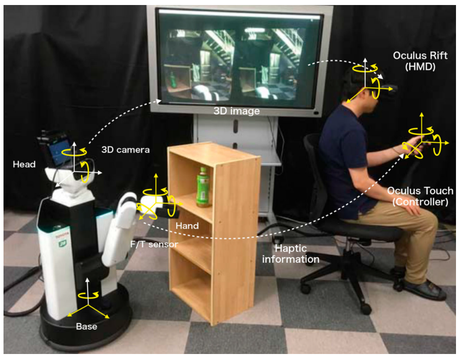


Figure 1.4: Overview of Toyota's Human Support Robot [40].

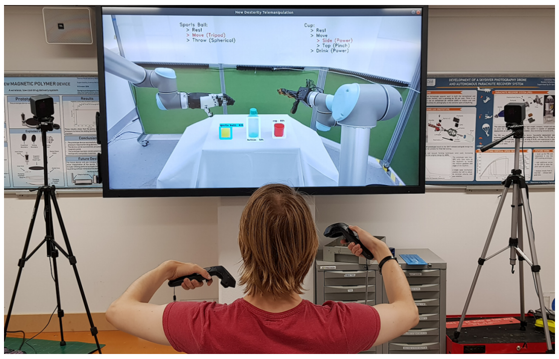


Figure 1.5: Bimanual telemanipulation of dual robot arm [38].

robot movement and vice versa was presented. It has also been showed that consumer grade VR tools can successfully be utilized for integration in teleoperation systems, allowing operators to demonstrate complex manipulation tasks [37] [38]. In [39], a VR gaming device is used to capture 6D hand movements to generate reference trajectories to a deep neural policy network for controlling the robot's joint movements.

An important objective in the development of the proposed teleoperation system is that the robot can be controlled intuitively, in the sense that a human can perform tossing operations using the robot manipulator, without prior training. For Toyota's Human Support Robot, an intuitive teleoperation system has been developed (see Figure 1.4), to explore the possibility in remote assistance of daily activities in domestic environments [40]. In [38], a dual robot arm is presented that can be controlled intuitively through a commercial VR interface. A novel approach for intuitive telemanipulation was presented in [41]. Here, an Augmented Reality (AR) interface is developed to assist the operator during operation, which increases situation awareness. Also, the mapping of the input device adjusts to the viewpoint from which the operator controls the robot. In [42], another approach is followed to achieve intuitive teleoperation: here the focus is on keeping the similarity of the configuration between the leader and follower arms, instead of only controlling the position of the manipulator's end-effector in its workspace. The examples above show that human operators are very well capable of performing precise, agile and dexterous tasks using teleoperation systems.

1.3 Problem Definition

In order to keep up with the growing demand of parcel delivery, the throughput of parcels in logistic processes has to be increased. Many tasks in logistics processes still require human operators, because these processes cannot yet be replaced by robots without reducing throughput and flexibility. However, if existing robots are used by their full potential, some of these manual tasks could be automated and thereby increase throughput, flexibility and costs.

The proposed solution is to make a transition in the way robots handle parcels. Instead of relatively slow pick-and-place operations, parcels could be handled more effectively using human-like pick-and-toss operations. However, deriving a model that can accurately predict the trajectory and final state of arbitrary packages that are tossed on a conveyor belt, is challenging.

By looking how a task is done, and by performing these tasks repeatedly, humans can often learn how to do it faster and more efficient. Robotic manipulation can be improved in a similar fashion. Therefore, the proposed method is to exploit and explore human intuition to control existing robots. Complex tasks require complex decision-making, and dexterous manipulation of objects.

Through teleoperation, cognitive skills and muscle motor skills of human operators can be utilized. By implementing a setup that is controlled by human hand motion, an intuitive and user-friendly interface is provided. Instead of requiring knowledge of external control hardware, the user is enabled to control the robot in a way that is already familiar to the user. Teleoperation allows the operator to perform motions intuitively, while conveniently controlling all robot degrees of freedom (DOFs) simultaneously. Introducing an advanced teleoperated robot could bridge the gap between the human manipulation skills and dynamic robot manipulation tasks. Yet, to the best of the author's knowledge, there is no such system available that is dedicated to human teleoperated tossing.

The development of this teleoperated robotic system involves the challenge of creating an intuitive mapping from the operator's hand to the robot's end-effector, while ensuring stable and safe control despite time delays, during free motion and interaction. This way of intuitive robot control may provide new insights towards the development of autonomous robot tossing solutions. By analyzing data from robotic tossing demonstrations, the hypothesis is that useful information can be acquired for modeling, perception, and control development for robotic tossing. Additionally, exploring new tossing strategies to see which ways of robotic tossing are the most effective is made easier. If the result of a toss can be repeated, leading to similar results, the outcome of a toss becomes predictable. By recording the control inputs of various tossing motions, and repeating these using the robot hardware, advances in our understanding in the predictability of tosses can be made. The primary research objective of this work is formulated as:

Develop a teleoperation system that enables operators to perform tossing demonstrations through hand motion, such that the industrial applicability of robot tossing can be explored.

The research objective of this project is divided into two sub-goals that must be met in order to achieve the primary goal, these are given by:

- *Develop a teleoperation system that enables operators to control the motion of a robot manipulator and activation of the suction gripper through hand motion.*
- *Quantify the variability in tossing outcomes resulting from autonomously repeating the demonstrated tossing motion, potentially introducing controlled perturbations, to explore the predictability of robot tossing.*

1.4 Thesis Outline

This thesis is structured as follows. Before diving into the main contributions of this research, Chapter 2 gives an overview of the relevant background information and early design decisions in the development of the teleoperation system. In this chapter, an industrial use case for robotic tossing is presented, together with a description of the Vanderlande Innovation Lab at the TU/e campus, where the system will be implemented and tested. Moreover, the term *teleoperation* is detailed, after which the system's requirements are presented. Then, the system's hardware and software components are introduced, the multibody dynamics notation used throughout this work is presented, and QP control is introduced. In Chapter 3, the design and implementation of the teleoperation system is discussed. This involves the hardware setup, communication channels and a discussion about how the system is controlled by the operator. Furthermore, the teleoperation controller algorithm is detailed, and the QP control approach is presented. Upon successful implementation of the proposed system, Chapter 4 demonstrates the robot's agility and dexterity through examples of what an operator can do using the system, particularly focused on tossing. Moreover, the predictability of robot tossing is tested through experiments. Finally, another set of experiments is done to explore how uncertainties in box picking affect the predictability of a toss. In Chapter 5 the conclusions and recommendations of this work are presented.

Chapter 2

Preliminaries

This chapter provides a review of the relevant background information for this research. First, an introduction into teleoperation is given, see Section 2.1. Then, in Section 2.2, an overview of the multibody dynamics notation used throughout this work is given. Finally, Section 2.3 covers an introduction into quadratic programming control.

2.1 Introduction to Teleoperation

Teleoperation, which literally means operations at a distance, generally refers to a remote manipulator that attempts to track the operator's input [43]. Hereby, the human operator is responsible for any high-level planning or cognitive decision of the execution of a task, while the robot is responsible for the task's mechanical execution. Essentially, in teleoperation, the brain is distant from the body. Teleoperation is beneficial in situations where a robot can provide an advantage in performing (a) physical task(s), while human cognitive skills or muscle motor skills are required. Therefore, teleoperation may be imposed to demonstrate dexterous motion trajectories, or scaling to very large or small environments. Teleoperation is also often found in hazardous environments, for example to rule out health risks while performing operations in radioactive, very hot, or explosive environment, by doing it remotely [43]. Even doctors may use teleoperation to perform minimally invasive surgery, to combine the knowledge of a surgeon to perform delicate operations with the precision of robotic devices [44]. In this work, a teleoperation system is developed to perform human demonstrated tossing, using a robot manipulator. This allows for quick generation of tossing motion trajectories, while using human cognition to toss objects towards desired locations. A sketch of the proposed teleoperation system is depicted in Figure 2.1. The device that is used by the human operator to control the remote robot, is often referred to as the leader device, while the remote robot that is controlled using the leader device, is referred to as the follower device. The teleoperation system, as a whole, is referred to as a leader-follower system. In older literature, teleoperation systems have often been referred to as master-slave systems, which was first used in 1904 [45]. Due to its association with slavery, use of this terminology has become a subject of controversy. Alternative terms that indicate the same are: boss-worker, parent-child, and controller-responder.

Often, teleoperation systems are conceptually split into two sites: the local site and remote site. At the local site, there are the human operator(s) and the input device(s) to send control commands. The remote site contains the robot and the required control elements. The sites are connected by a communication interface, through which control or communication signals can be transferred. The physical separation between the two sites may be very large, for example in the case of unmanned space missions. In the proposed teleoperation system, the physical separation between the hardware is relatively small, since the human operator and the robot will occupy the same room. This allows the operator to receive direct visual feedback of the system. We refer to this as collocated teleoperation. To provide direct control, the human operator holds the leader device,

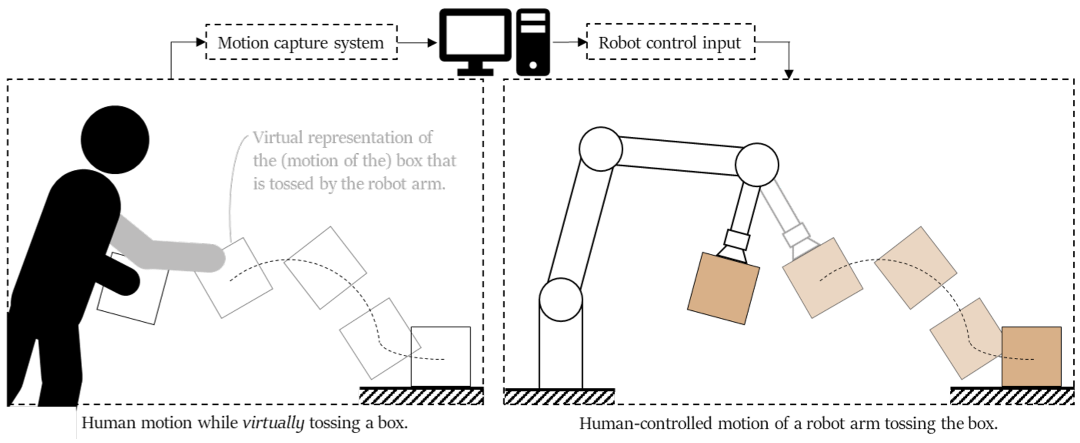

Motion capture system
Robot control input
Virtual representation of the (motion of the) box that is tossed by the robot arm.
Human motion while *virtually* tossing a box.
Human-controlled motion of a robot arm tossing the box.

Figure 2.1: Unilateral telemanipulation system for tossing by human motion.

while the follower device is programmed to follow its motions. Commonly, the leader device is a joystick, keyboard, ergonomic controller, or even a kinematic replica of the follower, to provide an intuitive interface. In some teleoperation systems, the follower device does not only measure motions, but also forces. This allows to provide force feedback to the user, through the leader device. Such systems are called bilateral teleoperation systems [35]. The ultimate goal of leader-follower systems is often to give the user the ability to manipulate the remote environment, while perceiving the environment as if encountered directly, this goal is referred to as telepresence. In unilateral teleoperation, which means that the operator does not receive feedback from the leader device, it is often more challenging to reach high levels of telepresence. The operator is responsible for the execution of the task, therefore, the teleoperation system should be designed in such a way that it becomes intuitive for the operator to perform manipulations. A good understanding of intuitivity and how this can be enhanced may help the development of a teleoperation system that is suitable for performing tossing operations. Therefore, a study into intuitivity is presented in Appendix B.

2.2 Multibody Dynamics Notation

The multibody dynamics notation as proposed in [46] is used throughout this work. For reference, the notations from [46] are presented here. First, the rigid body pose notation, expressed using coordinate frames, is described in Section 2.2.1. Then, the formulation to describe the velocity and acceleration of these coordinate frames is presented in Section 2.2.2. Finally, the notation to describe the position tasks for robot manipulators in robot control is presented in Section 2.2.3.

2.2.1 Coordinate Frames and Pose Notation

To represent the position and orientation of a rigid body in 3D space, by standard practice, coordinate frames are used. A coordinate frame is defined by an origin and an orientation frame. Coordinate frames are denoted by a capital letter, for example, for frame A, its origin is written as $\mathbf{o}_A$ and its orientation frame as $[A]$. Formally, this is written as $A = (\mathbf{o}_A, [A])$. Coordinate frames can move through space in time with respect to other coordinate frames. Typically, they are assigned to rigid bodies to describe their position and orientation in space, often located at the geometrical center or center of mass. To track the pose of a rigid body in time, it can be expressed in a reference frame.

For a point $\mathbf{p}$ with respect to A, its coordinates are written as the coordinate vector ${}^{A}\mathbf{p}$, which represents the 3D geometric vector $\vec{r}_{\mathbf{o}_A,\mathbf{p}}$. This vector starts at the origin $\mathbf{o}_A$, is pointing towards

$\mathbf{p}$, and is expressed in terms of $[A]$. Formally, this can be written as

$$
{}^{A}\mathbf{p} := \begin{bmatrix} \vec{r}_{\mathbf{o}_A,\mathbf{p}} \cdot \vec{x}_A \\ \vec{r}_{\mathbf{o}_A,\mathbf{p}} \cdot \vec{y}_A \\ \vec{r}_{\mathbf{o}_A,\mathbf{p}} \cdot \vec{z}_A \end{bmatrix} \in \mathbb{R}^3, \tag{2.1}
$$

where $\cdot$ denotes the scalar product and $\vec{x}_A, \vec{y}_A$, and $\vec{z}_A$ are mutually orthogonal unit vectors defining the orientation frame $[A]$. Given the frames A and B, the coordinate transformation from $[B]$ to $[A]$ can be described by using the rotation matrix ${}^{A}\mathbf{R}_B \in \mathbf{SO}(3)$. Note that this coordinate transformation is independent of the positions of $\mathbf{o}_A$ and $\mathbf{o}_B$, and thus only depends on the orientation. To describe both the position and orientation, hereafter referred to as the pose, of frame B relative to frame A, the 4×4 homogeneous transformation matrix ${}^{A}\mathbf{H}_B \in \mathbf{SE}(3)$ is used, which is given by (2.2).

$$
{}^{A}\mathbf{H}_B := \begin{bmatrix} {}^{A}\mathbf{R}_B & {}^{A}\mathbf{o}_B \\ \mathbf{0} & 1 \end{bmatrix}. \tag{2.2}
$$

Using the homogeneous transformation matrix above, any given point expressed in one frame, can be expressed in another frame. This, for example, is expressed by

$$
{}^{A}\overline{\mathbf{p}} = {}^{A}\mathbf{H}_B {}^{B}\overline{\mathbf{p}}, \tag{2.3}
$$

where frame B, is expressed in frame A, given that ${}^{A}\overline{\mathbf{p}}$ and ${}^{B}\overline{\mathbf{p}}$ denote the homogeneous representation of ${}^{A}\mathbf{p}$ and ${}^{B}\mathbf{p}$ as

$$
{}^{A}\overline{\mathbf{p}} := \begin{bmatrix} {}^{A}\mathbf{p} \\ 1 \end{bmatrix}, \quad {}^{B}\overline{\mathbf{p}} := \begin{bmatrix} {}^{B}\mathbf{p} \\ 1 \end{bmatrix} \tag{2.4}
$$

2.2.2 Rigid-Body Velocity Notation

To obtain the velocities of points and origins with respect to a reference frame, differentiation is used as

$$
{}^{A}\dot{\mathbf{p}} = \frac{d}{dt}\left({}^{A}\mathbf{p}\right), \quad {}^{A}\dot{\mathbf{o}}_B = \frac{d}{dt}\left({}^{A}\mathbf{o}_B\right). \tag{2.5}
$$

Similarly, the rotation matrix ${}^{A}\mathbf{R}_B$ and the homogeneous transformation matrix ${}^{A}\mathbf{H}_B$ can be differentiated to obtain ${}^{A}\dot{\mathbf{R}}_B$ and ${}^{A}\dot{\mathbf{H}}_B$, respectively. In order to obtain a more compact representation of ${}^{A}\dot{\mathbf{H}}_B$, it can be pre- or postmultiplied with the inverse of ${}^{A}\mathbf{H}_B$, hereby obtaining an element of $\mathfrak{se}(3)$, which are called twists. *Pre*-multiplying leads to

$$
\begin{aligned}
{}^{A}\mathbf{H}_B^{-1}{}^{A}\dot{\mathbf{H}}_B &= \begin{bmatrix} {}^{A}\mathbf{R}_B^T & -{}^{A}\mathbf{R}_B^{T}{}^{A}\mathbf{o}_B \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} {}^{A}\dot{\mathbf{R}}_B & {}^{A}\dot{\mathbf{o}}_B \\ \mathbf{0} & 0 \end{bmatrix} \\
&= \begin{bmatrix} {}^{A}\mathbf{R}_B^{T}{}^{A}\dot{\mathbf{R}}_B & {}^{A}\mathbf{R}_B^{T}{}^{A}\dot{\mathbf{o}}_B \\ \mathbf{0} & 0 \end{bmatrix}.
\end{aligned} \tag{2.6}
$$

Since the term ${}^{A}\mathbf{R}_B^{T}{}^{A}\dot{\mathbf{R}}_B$ is skew-symmetric, it can be written as a vector in $\mathbb{R}^3$. This is done using the $(\cdot)^\wedge$ hat-operator which transforms a vector in $\mathbb{R}^3$ to a skew-symmetric matrix in $\mathfrak{so}(3)$, such that

$$
\boldsymbol{\omega}^\wedge = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}^\wedge := \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \in \mathfrak{so}(3). \tag{2.7}
$$

To this end, we define

$$
{}^{B}\boldsymbol{v}_{A,B} := {}^{A}\mathbf{R}_B^{T}{}^{A}\dot{\mathbf{o}}_B \tag{2.8}
$$

$$
{}^{B}\boldsymbol{\omega}_{A,B}^\wedge := {}^{A}\mathbf{R}_B^{T}{}^{A}\dot{\mathbf{R}}_B. \tag{2.9}
$$

From the two equations above, the *left trivialized* velocity of a frame B with respect to frame A, is given by

$$^{B}\mathbf{v}_{A,B} := \begin{bmatrix} ^{B}\boldsymbol{v}_{A,B} \\ ^{B}\boldsymbol{\omega}_{A,B} \end{bmatrix} \in \mathbb{R}^{6}. \tag{2.10}$$

2.2.3 Joint States and Jacobians

In the proposed teleoperation system, the UR10 serial robotic arm is controlled in three-dimensional Cartesian space, where only the robot's joints can be actuated. The coordinate frames and associated velocities and accelerations of the bodies of robotic manipulators, can be expressed in terms of their joint states and corresponding Jacobians. Jacobians are essential tools to express velocity and acceleration tasks in robot control.

Consider the 4DOF robot manipulator illustration in Figure 2.2. Here, the fixed inertial frame lays in the robot's base, denoted by U, and the end-effector frame is denoted by S. The end-effector pose S, can be expressed in terms of the joint positions of the manipulator given by $\mathbf{q} \in \mathbb{R}^n$, with n the number of joints in the manipulator. If the forward kinematics relating the joint positions to the coordinate frame S is denoted by h, we can say $S = h(\mathbf{q})$.

The velocity of frame S can be expressed with respect to the inertial frame U. Using the Jacobian, these velocities can also be calculated using the robot's joint velocities $\dot{\mathbf{q}} \in \mathbb{R}^n$. The left trivialized velocity of S, is given by

$$^{S}\mathbf{v}_{U,S} = {}^{S}\mathbf{J}_{U,S}(\mathbf{q})\dot{\mathbf{q}}, \tag{2.11}$$

where $^{S}\mathbf{J}_{U,S}(\mathbf{q})$ denotes the Jacobian relating the velocity of S with respect to U expressed in frame S. The Jacobian is dependent on the joint positions $\mathbf{q}$. The associated acceleration is computed by

$$^{S}\dot{\mathbf{v}}_{U,S} = {}^{S}\mathbf{J}_{U,S}(\mathbf{q})\ddot{\mathbf{q}} + {}^{S}\dot{\mathbf{J}}_{U,S}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}. \tag{2.12}$$

To compute the right trivialized velocity $^{U}\mathbf{v}_{U,S}$ and the associated accelerations, the term $^{S}\mathbf{J}_{U,S}(\mathbf{q})$ in (2.11) and (2.12) is replaced by $^{U}\mathbf{J}_{U,S}(\mathbf{q})$, respectively.

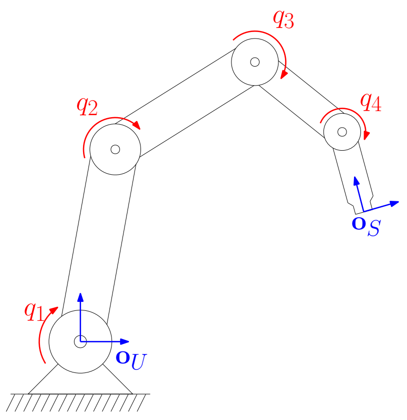


Figure 2.2: Schematic illustration of a 4DOF robot manipulator.

2.3 Quadratic Programming Control

A quadratic program (QP) is an optimization problem with a quadratic objective function and linear constraints. QP control is based on solving quadratic programs every control time step, and thereby calculating the optimal solution for a certain task. In this project, we use a QP control framework to control the robot manipulator's motion, see Section 3.1.4. Therefore, in this section, we briefly introduce QP control. The final QP controller approach is presented in Section 3.3.

2.3.1 Quadratic Programming

A general QP problem, see Equation (2.13), consists of a quadratic cost function (2.13a) with linear equality constraints (2.13b) and linear inequality constraints (2.13c), see

$$\min_{\mathbf{x}} \quad \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{c}^\top \mathbf{x} \tag{2.13a}$$

$$\text{s.t.} \quad \mathbf{A}_1\mathbf{x} = \mathbf{b}_1 \tag{2.13b}$$

$$\mathbf{A}_2\mathbf{x} \geq \mathbf{b}_2. \tag{2.13c}$$

Here, $\mathbf{x} \in \mathbb{R}^n$ denotes the vector of optimization variables, $\mathbf{Q} \in \mathbb{R}^{n\times n}$ denotes the symmetric Hessian matrix, $\mathbf{c} \in \mathbb{R}^n$ is a vector of constants, $\mathbf{A}_1 \in \mathbb{R}^{k\times n}$ and $\mathbf{b}_1 \in \mathbb{R}^k$ define the k equality constraints, and $\mathbf{A}_2 \in \mathbb{R}^{\ell\times n}$ and $\mathbf{b}_2 \in \mathbb{R}^\ell$ the ℓ inequality constraints, n denotes the optimization variable state dimension [47]. Often, the cost function of a QP problem (2.13a) is written in the form of (2.14), by using $\mathbf{E} = \mathbf{Q}^T\mathbf{Q}$ and $\mathbf{c} = \mathbf{Q}^T\mathbf{b}$.

$$\min_{\mathbf{x}} \quad \frac{1}{2}\|\mathbf{E}\mathbf{x} - \mathbf{b}\|^2 \tag{2.14}$$

The goal of the QP problem is to find the values of $\mathbf{x}$, for which the cost function is optimized. To ensure that there exists a unique optimal solution for the optimization problem, that can be found quickly, the QP problem must be convex.

2.3.2 Task-Based Constrained QP Control

Quadratic programming can also be used to control robots. In such controllers, a QP is instantiated at every control time step, minimizing the error of multiple desired tasks under all physical and structural constraints of the robot. In robotic applications, QP control is often praised because it allows to perform multiple tasks simultaneously, with relative importance between tasks, ordered via a weighted approach. QP control is particularly useful for humanoids (e.g. balancing while reaching objects with multiple end-effectors), but robot manipulators can also favor from it, for example while performing end-effector tasks while avoiding collision or joint limit violations.

In a QP controller, multiple simultaneous objectives can be used to minimize a cost function. The objectives of a robot's motion are expressed in terms of features of the system, where a feature refers to any function f that describes a motion. For example, the position of the hand of a humanoid, or the configuration of a door (opening angle), or the position/orientation of a drone, etc. The value of each objective $\left\|\ddot{f} - \ddot{f}^d\right\|$ measures the difference between the current and desired accelerations, respectively, $\ddot{f}$ and $\ddot{f}^d$. For a number $\mathcal{N}$ of simultaneous objectives $(f_1, \ldots, f_{\mathcal{N}})$, the quadratic cost function to minimize in the QP is defined as

$$c_{\mathbf{q},\dot{\mathbf{q}}}(\ddot{\mathbf{q}}) = \sum_{i=1}^{\mathcal{N}} w_f \left\|\ddot{f}_i - \ddot{f}_{d_i}\right\|^2, \tag{2.15}$$

where w_f are the relative weights of the objectives that are tuned depending on which objective is favored, depending on the observed behavior resulting from that choice (e.g., in case of a humanoid, falling over would suggest increasing a center of mass objective weight).

In the proposed teleoperation system, the robot's current and desired end-effector pose will both be described in task-space. Therefore, the control strategy will use the current state of the robot in task-space for feedback, this is referred to as task-based control. Based on the desired end-effector pose, forward kinematics are used to calculate the robot's state $\mathbf{q}$ in joint space, which is fed into the QP controller. To solve the QP problem on velocity or acceleration level, the constraints have to be linear in $\dot{\mathbf{q}}$ or $\ddot{\mathbf{q}}$, respectively. Because the dynamics of a fixed-base robot manipulator are linear in $\ddot{\mathbf{q}}$, while being nonlinear in $\mathbf{q}$ and $\dot{\mathbf{q}}$, and the robot dynamics have to be taken into account while solving the QP problem, an acceleration based formulation is preferred. Moreover, since joint accelerations allow for jumps in their control commands, minimizing the QP problem for $\ddot{\mathbf{q}}$ also leads to smoother robot motion, compared to a velocity level QP controller. A general task-based QP formulation with the joint accelerations as minimization variable is given by 2.16.

$$\min_{\ddot{\mathbf{q}}} \quad c_{\mathbf{q},\dot{\mathbf{q}}}(\ddot{\mathbf{q}}) \tag{2.16a}$$

$$\text{s.t.} \quad \mathbf{A}_1\ddot{\mathbf{q}} = \mathbf{c}_2 \tag{2.16b}$$

$$\mathbf{A}_2\ddot{\mathbf{q}} \leq \mathbf{c}_2 \tag{2.16c}$$

where $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ denote the robot's joint velocities and accelerations, respectively. While solving the QP problem, the weighted sum of the task errors $c_{\mathbf{q},\dot{\mathbf{q}}}(\ddot{\mathbf{q}})$ should express stabilized error dynamics.

2.4 Introduction to Finite-State Machines

In the QP framework that we use to control the robot, see Section 3.1.4, a finite-state machine (FSM) controller implementation is facilitated. For this reason, we couple the proposed QP controller with a finite-state machine to automatically perform a sequence of tasks using the robot manipulator.

A finite-state machine (FSM) is an abstract machine that, at a given time, is in one state from a finite set of states [48]. In response to internal or external conditions, the machine's state can change. Such a change is referred to as a transition. A list of modes, with one initial mode, and a definition of the conditions that trigger each transition, together define an FSM. Throughout this work, the word state has been referred to as the continuous state of the dynamics (i.e. $q, \dot{q}, \ddot{q}$). To avoid confusion of terms, in the remainder of this work, we refer to an FSM *state* as a *mode*. FSMs are particularly well-suited to implement repetitive or sequential robotic tasks, due to the simplicity and the composability it offers. For example, to control the locomotion of a humanoid, its motion can be divided into different phases, or *modes*. A finite-state machine handles the transitions between these modes when transition conditions are realized, for example, foot landed, COM shifted, grasp established, grasp released, contact established, contact broken, etc. See Figure 2.3 for an example from [48], showing the building blocks of locomotion of a humanoid robot.

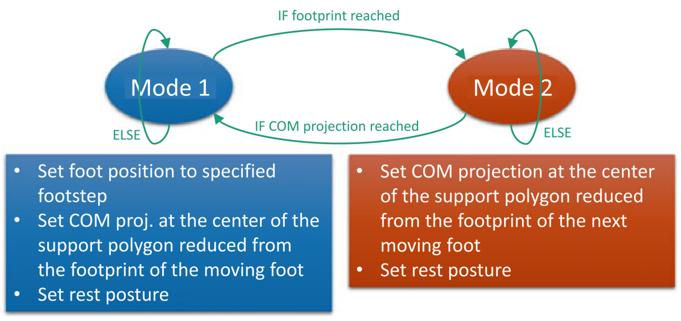

IF footprint reached
Mode 1
Mode 2
ELSE
IF COM projection reached
ELSE
• Set foot position to specified footstep
• Set COM proj. at the center of the support polygon reduced from the footprint of the moving foot
• Set rest posture
• Set COM projection at the center of the support polygon reduced from the footprint of the next moving foot
• Set rest posture

Figure 2.3: FSM example from [48] containing two modes for cyclic locomotion of a humanoid robot, alternating left foot and right foot steps.

Chapter 3

System Design and Implementation

In this chapter, the design and implementation of the teleoperation system is presented. In Section 3.1, the chosen hardware, corresponding setup, and software components are introduced. After this, the system's communication framework, control input generation, and controller implementation are detailed in Section 3.2. Finally, the associated QP control approach is presented in Section 3.3.

3.1 Teleoperation System Outline

In this section, the robot manipulator and leader device, with the corresponding hardware setup, are presented. Moreover, the software packages on which the proposed teleoperation system relies are introduced. A schematic view of the proposed teleoperation system is depicted in Figure 3.1. Prior to the system design, various design considerations were contemplated, for this discussion, please refer to Appendix C. Additionally, the system requirements are stated in Appendix D.

3.1.1 Robot Manipulator

There are two robot manipulators available for the teleoperation system: Franka Emika's *Panda* and Universal Robots' *UR10*. Both robot manipulators are equipped with Smart Robotics' custom vacuum gripper. The hardware specifications of both robots are stated in Appendix E.1 and E.2. In a comparison between the two available robot manipulators, the main differences were the number of degrees of freedom, payload, and maximum end-effector velocity. While the Panda has a more anthropomorphic design than the UR10, which enhances intuitivity, its kinematic redundancy also significantly complicates the controller design. Moreover, its payload (max. 3

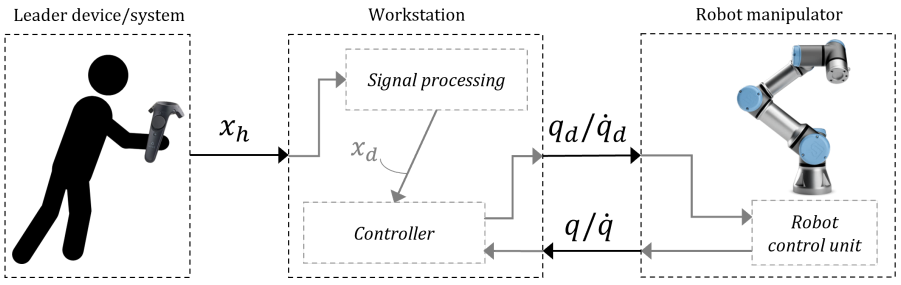


Figure 3.1: Unilateral teleoperation system that senses hand movements and uses these as input signal x_h to the workstation. The hand motion is converted to a reference signal x_d for the controller, which calculates desired joint control inputs for the robot manipulator.

kg) and maximum end-effector velocity (max. 1.7 m/s) are low compared to the UR10 (max. 10 kg and max. 3 m/s). Based on inertial motion capture measurements using an XSens suit, an analysis of human tossing motions was made. This analysis revealed that the human hand typically achieves velocities of 2 to 4 m/s, which the Panda is unable to reach. For these reasons, the UR10 was chosen to be utilized in the proposed teleoperation setup. For a more detailed comparison and choice substantiation, please refer to Appendix E.3. In the remainder of this section, the UR10 and SR is presented into more detail.

Universal Robots' UR10

The UR10, see Figure 3.2, is a collaborative robot manufactured by Universal Robots [49]. Collaborative robots, also referred to as cobots or co-robots, are robots intended for environments where humans and robots share space or are in proximity, allowing for safe and direct human-robot interaction [50]. Traditional industrial robots function separated from their human co-workers, often with fences or cages to guarantee safety. Cobots are designed to handle shared payload, and to function safely without conventional protective measures. To ensure safe human-robot interaction, cobots rely on lightweight construction materials, rounded edges, and inherent limitation of speed and force, and/or sensors and software that ensure safe behavior. The cobots provide a benefit in industrial settings such as logistics warehouses, since the advantages that robots and humans have can be exploited simultaneously [51]. Human-robot collaboration may be preferable to purely robotic operations, by leveraging the advantages of both humans and robots. For experimental research settings, this has already shown to be true [52]. Because of this, using the UR10 cobot as a follower device in a teleoperation system, to teach tossing operations in logistic tasks, seems to be a logical choice.

The UR10 is a 6 degrees of freedom (DOF) arm, which consist of seven bodies: the shoulder, upper arm, forearm, three wrists, and (in this case) a suction gripper, see Figure 3.2. Each joint is equipped with high resolution joint position and velocity encoders, which allow for accurate pose control [49]. Because the robot allows for the installation of external grippers and has various configurable inputs and outputs to connect with accessories, the robot can be employed for a wide range of tasks. Moreover, the UR10 has a Power and Force Limiting (PFL) functionality, which is a safety measure that manages that a protective stop is issued, whenever an external force leads to an exceedance of a preset limit value. Moreover, the UR10 makes use of space limiting which is used to define any geometric shape which may be used to limit robot motion within the defined space, or preventing the robot from entering the defined space. The UR10 is controlled by a control box, where internal and external communication occurs at a maximum frequency of 125 Hz. The control box can be programmed using its teach pendant, or via a Linux PC through an Ethernet connection. A PC can be used to run external applications on the UR robots, by

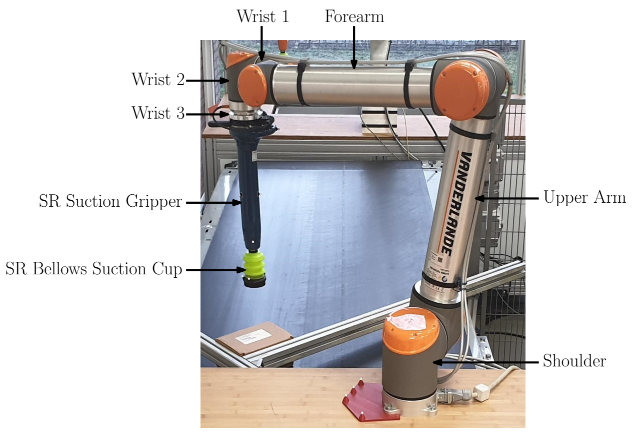


Figure 3.2: UR10 equipped with Smart Robotics (SR) vacuum gripper.

communicating inputs and outputs through Real-Time Data Exchange (RTDE) at 125 Hz. The RTDE interface is the standard interface of UR arms to synchronize external applications with the UR controller [53]. Another interface to run external applications on the robot is ROS UR, which is used to communicate joint states, tool center pose, and I/O's between the PC and robot. The latter interface is used to control the UR10 in the Vanderlande innovation lab.

Smart Robotics Gripper

The UR10 that is used in this project is equipped with the vacuum gripper of Smart Robotics, see Figure 3.2. This gripper is equipped with a Venturi ejector, which generates a partial vacuum inside the gripper, allowing to smoothly and securely handle a wide variety of objects. The gripper functions as the interface between the actuated robot and the manipulated object. The gripper is build from three main parts: the suction cup, tool arm, and the ejector. The bellows-type suction cup allows for picking a wide variety of objects, and due to its flexibility, the suction cup easily bends towards objects. Because of this, accurate pose estimation to pick objects is not required. The tool arm connects the robot arm to the suction cup, and is a rigid arm consisting of internal channels, which the air from the ejector flows through. The tool arm is also equipped with an air pressure sensor, using which it can be detected if the suction cup is attached to an object. One can imagine that it is of great importance that the robot quickly detaches an object when the operator commands the robot to release it while tossing. For improved detachment of objects when these are released, the ejector is equipped with a blow-off functionality, which forces the object to be released as fast as possible. Without this functionality, the vacuum results in a delayed detachment of objects. Using the gripper, objects with a mass up to 8 kilograms can be lifted.

3.1.2 Leader Device

The teleoperation system required a leader device that enables humans to effectively and intuitively control the robot's motion, such that robotic tossing operations can be demonstrated. To this end, the device must be capable of 6DOF motion sensing (3 translational directions, and 3 rotational directions), and needs to have a possibility to control the suction gripper, for activation of the vacuum and blow-off functionalities.

First, the state-of-the-art in motion capture (hereafter referred to as mocap) technology is analyzed, see Appendix F.1. For each mocap technology, a market review into commercially available systems that can potentially be employed in the teleoperation system is made. Based to this a market review, see Appendix F.2, a number of good candidate devices were found. For example, vision based mocap technology of OptiTrack[1], inertial based mocap of XSens[2], and a combination of mocap technologies by Noitom[3]. Also commercial gaming devices are considered, such as VR controllers from Oculus[4] and HTC Vive[5]. As the popularity of motion capture technology grows, mainly due to its application in gaming, the cost of good VR equipment falls. Because of this, off-the-shelf VR devices are providing a feasible alternative for mocap devices in the context of robotics. Because of its good portability, ergonomic design, integratability, and good price-quality ratio in terms of tracking performance, The HTC Vive Pro kit system was chosen as leader device in the proposed teleoperation system. In Figure 3.3, a picture of the HTC Vive Pro kit is depicted.

The pose of the VR controller is estimated at a 120 Hz refresh rate, with sub-millimeter accuracy [54]. The average position's standard deviation for pose estimation of the static (not moving) device was found to be 0.242 mm, and its dynamic tracking accuracy varies in the millimeter range. However, experimental results have shown that the tracking error can become up to a meter [54]. In [55], experimental research has shown that the position tracking accuracy of the

[1]See `https://www.optitrack.com/`.
[2]See `https://www.xsens.com/products/xsens-gloves-by-manus`.
[3]See `https://www.noitom.com/hi5-vr-glove`.
[4]See `https://www.oculus.com/quest-2/`.
[5]See `https://www.vive.com/us/product/vive-pro/`.

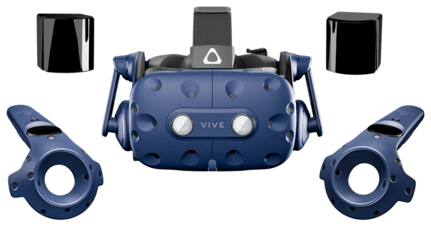


Figure 3.3: HTC Vive Pro Kit [56].

system is few millimeters, when two lighthouses are used. For detailed information about how the Vive system works, and its technical specifications, please refer to Appendix G.

There is a good deal of literature available where commercial VR (gaming) controllers are used in research projects for (precise) control. In [37] the Vive VR controller has been used as leader device in a teleoperation system with a Franka Emika Panda robot. In [57], a method targeting repair in space using a cobot, also controlled by a HTC Vive device, is presented. And in [58], an ABB YuMi cobot was used as an extension for surgical tools, while also being controlled by a VR system. Moreover, for VR controllers from multiple vendors, like the one from HTC Vive, great software development support is available through OpenVR [59]. OpenVR, which is developed by Valve Corporation, provides an extensive software development kit (SDK) and application programming interface (API). OpenVR provides great assistance to engineers that are inexperienced with software development, like the author of this work.

3.1.3 Hardware Setup

The proposed teleoperation system is operated by a human, and employs three hardware systems: the Vive VR Pro kit, a workstation (HP EliteDesk 800 G5 TWR, Core i7 vPro 9th Gen) running Linux with Ubuntu 20.04 as operating system, and the UR10 equipped with a vacuum gripper. Figure 3.4 shows a schematic overview of the proposed teleoperation setup in the Vanderlande Lab at the TU/e campus, see Appendix A.

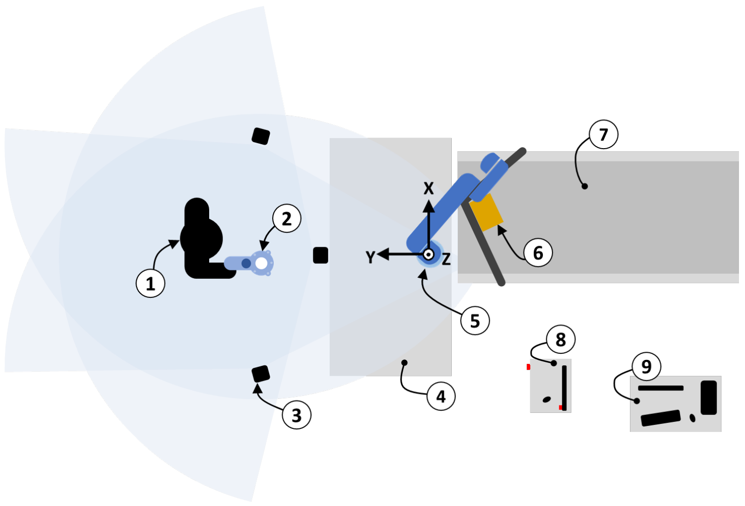


Figure 3.4: Colocated unilateral teleoperation setup with: an operator (1) that holds a Vive VR controller (2) which is tracked by three basestations (3). The UR10 (5) is mounted on a test bench (4), and located in front of a conveyor (7) with a parcel (6). Next to the setup, the robot control box (8) and workstation (9) are depicted. The system's inertial frame is located at the robot's base.

The proposed teleoperation system enables humans to control the UR10's motion, specifically for tossing, using Vive's VR controller. While controlling the teleoperation system, the operator does not receive haptic feedback, and the operator is located in the same room as the robot. This is referred to as unilateral collocated teleoperation. In the final setup, it is made sure that the operator is always close to the robot manipulator, while also being able to receive visual feedback, without any communication delay, and from a safe distance. The operator's viewpoint, as shown in Figure 3.4, is chosen such that the operator has clear overview of the robot's surroundings while performing tossing demonstrations. Moreover, since the operator will demonstrate robotic tossing aiming for the conveyor belt, this viewpoint seems convenient.

3.1.4 Software

The proposed teleoperation system will communicate and be controlled through a central PC (HP EliteDesk 800 G5 TWR, Core i7 vPro 9th Gen), running Linux with Ubuntu 20.04 as operating system. The UR10 in the Vanderlande Innovation Lab at the TU/e campus has already been interfaced with the Linux PC for through controller software that goes by the name of mc_rtc [60]. The controller for the teleoperation system is written and implemented using mc_rtc. To use the Vive VR controller as leader device for teleoperation, an interface between the VR hardware and mc_rtc is made through OpenVR. OpenVR serves as an interface between the VR hardware and VR software, and is implemented by SteamVR. As part of this research work, an interface has to be made between mc_rtc and the VR controller device. To this end, the software packages that are involved in this are presented.

QP Control via mc_rtc Software

The proposed teleoperation system is controlled through the mc_rtc QP control framework. This controller framework uses QP to calculate the required joint accelerations to control the robot's end-effector towards a desired pose. mc_rtc is developed by Joint Robotics Laboratory (JRL) of CNRS-AIST [61], for simulation and robot control systems. The mc_rtc framework provides an interface which can be used for real-time control of simulated and real robotic systems. Moreover, the close collaboration between CRNS and the I.AM project provides an advantage in the development of the proposed teleoperation setup. By providing a real or estimated state of a given robot state (joints' values, sensor readings, etc.), mc_rtc can compute the commands to control the robot towards the desired robot's state. Moreover, the mc_rtc framework uses tools that are built around the Robot Operating System (ROS) [62], these tools provide extra functionalities, such as an interface to visualize the robot's surfaces within RViz [63]. In this work, the controller implementation is written in C++. For more information about the control framework, please refer to its tutorials, API documentation, and JSON/YAML documentation[6]. Additional information can be found on its GitHub repository [64].

SteamVR and OpenVR

VR controllers from multiple vendors, like the one from Vive, run on SteamVR to provide an interface for gaming using the VR hardware. SteamVR is a tool of Steam [65], which provides runtime to access the pose data and I/O input of each component of the VR kit. OpenVR is an Application Programming Interface (API) and runtime which allows for accessing data that is streamed by SteamVR [59]. The OpenVR C++ library provides a way to connect and interact with VR hardware without relying on a specific hardware vendor's SDK. OpenVR is implemented as a set of C++ interface classes with virtual functions, these are used in the controller implementation of mc_rtc, which is also written in C++. For additional information, please refer to [59].

[6]See: `https://jrl-umi3218.github.io/mc_rtc/`.

3.2 Teleoperation System Architecture

With the hardware components and the software packages described in Section 3.1, in this section, the system architecture is presented. This includes a detailed description of the communication framework. Moreover, the methodology in the teleoperation controller, and the control algorithm are presented. Furthermore, the QP control approach is presented.

3.2.1 Communication Framework

Each component in the proposed teleoperation system is connected and interacts with each other. In this section, these interactions are presented. Figure 3.5 shows a schematic overview of the basic interactions in the teleoperation system, where the interaction between components is indicated by arrows.

As described in Section 3.1.3, the human operator only receives live visual feedback by looking at the robot manipulator while controlling it. Moreover, the operator controls the robot's motion and suction gripper using the leader device. Also, the workstation is operated by a human operator, which can be used to change (parameter or controller) settings, start SteamVR, and run the controller. One of the main challenges of this work was to find a way to establish the communication between the commercial VR controller device and mc_rtc, such that the VR controller can be used as a leader device in the teleoperation setup. To establish communication between SteamVR and mc_rtc, the SteamVR runtime had to be accessed. To this end, the OpenVR Software Development Kit (SDK) was utilized. OpenVR (also developed by Valve) is an application programming interface (API) and runtime that allows access to VR hardware from multiple vendors [59], for which no specific knowledge about the hardware required. By writing the CMakeLists files such that these automatically install the appropriate OpenVR library on a dedicated location in the folder where the state files are accessed by the controller, the SteamVR runtime could be accessed directly from the mc_rtc.The pose of the VR controller device tracking and button data, that is operated by the operator, is captured in real-time at 120 Hz. Within mc_rtc, this pose and button data is used in C++ objects to generate a reference input for the QP controller and control the robot I/O's, such as switching the vacuum and blow-off in the gripper. Moreover, another C++ function is written to generate a reference input that contains the desired pose input, for which

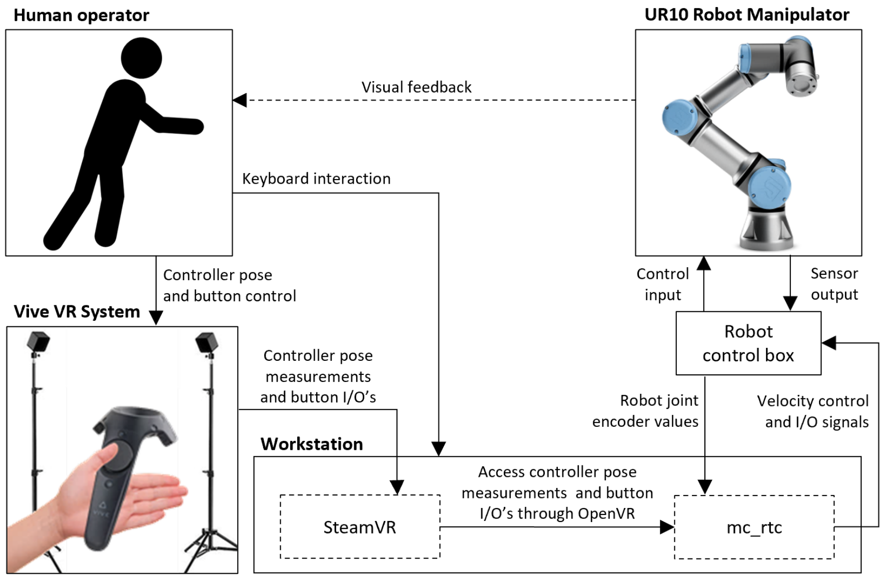

Human operator
UR10 Robot Manipulator
Visual feedback
Keyboard interaction
Controller pose and button control
Vive VR System
Control input
Sensor output
Robot control box
Controller pose measurements and button I/O's
Robot joint encoder values
Velocity control and I/O signals
Workstation
Access controller pose measurements and button I/O's through OpenVR
SteamVR
mc_rtc

Figure 3.5: Communication framework of the proposed teleoperation system. A human operator controls the robot's end-effector motion, and robot vacuum gripper through a Vive VR controller.

the desired joint accelerations are calculated by the QP implementation in mc_rtc. The joint accelerations are integrated to a velocity control signal, and sent together with the I/O signals to the robot's control box. The UR10 is controlled at a frequency of 125 Hz. The UR10 is equipped with encoders in each joint, which are used to measure the exact joint positions of the robot. Each time the QP controller is started, or the FSM mode is switched, the QP controller is initialized with the measured joint states of the robot. During control, the encoder values of the robot are not fed back into the QP controller, hence, the QP controller does not control the robot's motion through closed-loop control.

3.2.2 Reference Control Input Generation

With the communication established, the VR controller's button I/O's and pose measurements are used to generate a reference pose input signal for the QP controller. In this section, we discuss how the desired pose of the robot's end-effector is computed, and how the robot's suction gripper is controlled using the leader device.

Coordinate Frames

The first step in generating a reference input trajectory for the robot's end-effector using the VR controller, is to make sure the input (leader device) and output (robot) directions are aligned. The relevant coordinate frames that are referred to in this work, according to the hardware setup presented in Section 3.1.3, are schematically depicted in Figure 3.6. The inertial frame U, finds its origin at the UR10's base. Frame S is located in the center of the bottom of the robot's suction cup. Here, the robot manipulator makes contacts while gripping onto objects. Note that frame U and frame S are connected through the chain of links of the robot manipulator. The box' frame is denoted by B, which finds its origin at the center of mass of the box. Finally, the conveyor frame is denoted by C, which is represented by a contact surface with which the box (cuboid) can collide. The leader device frame, denoted by V, is measured with respect to Vive's global frame H. The origin and orientation of Frame H, are set during the calibration of the VR kit, which is done during the *room setup* in SteamVR[7].

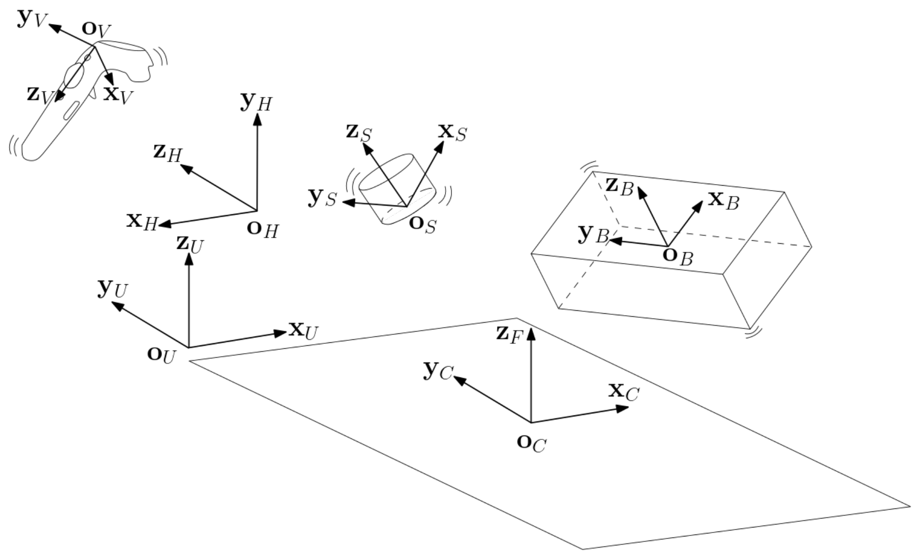


Figure 3.6: Illustration of the relevant frames in the teleoperation system. The inertial frame, located at the robot's base, is given by U. Additionaly, C denotes the conveyor frame, the suction cup frame is given by S, and the box frame is denoted by B. The pose of the VR controller is denoted by V, which is measures with respect to Vive's measurement frame H.

[7] For details on using the Vive devices, such as the room setup, see `https://www.vive.com/eu/support/vive-pro-hmd/`.

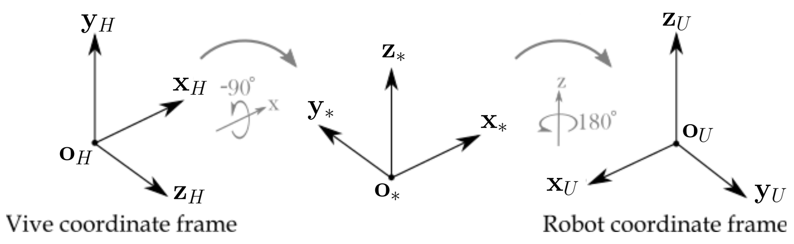


Figure 3.7: To align $[H]$ with frame $[U]$, such that $[H] = [U]$, H is rotated $-90°$ over the x-axis, and subsequently rotated $180°$ over the z-axis.

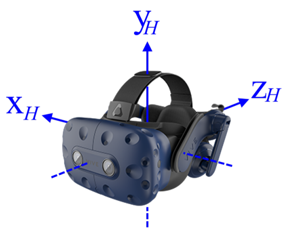
In each control step t_k, the pose of the leader device V is retrieved from SteamVR, through a 4×4 pose matrix. As the leader device pose V is measured in its own frame, denoted by H, this pose matrix is denoted by ${}^H\mathbf{H}_V$. The pose of the leader device ${}^H\mathbf{H}_V$ is retrieved through a 3×3 orientation matrix ${}^H\mathbf{R}_V$ and a 3×1 position vector ${}^H\mathbf{o}_V$. At a time instance t, the pose of the leader device is denoted by

$$ {}^H\mathbf{H}_V(t) = \begin{bmatrix} {}^H\mathbf{R}_V(t) & {}^H\mathbf{o}_V(t) \\ \mathbf{0} & 1 \end{bmatrix}. \tag{3.1} $$

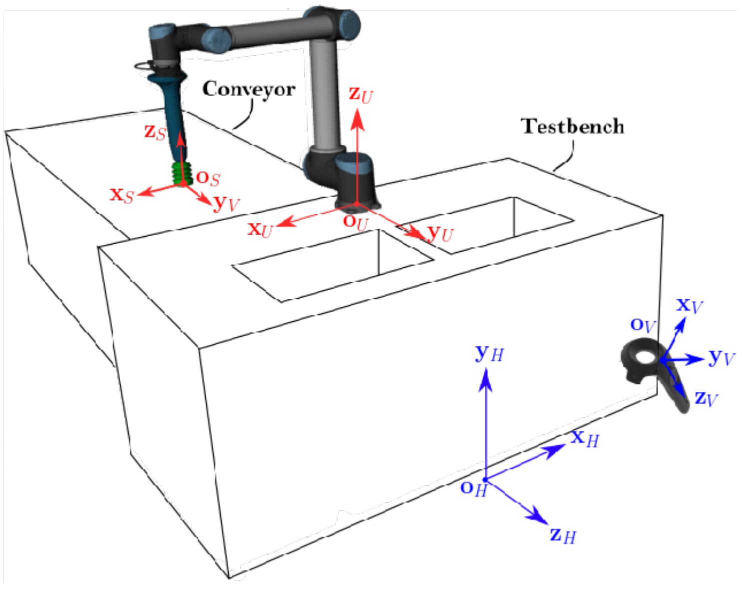
In order to align the orientation of Vive's inertial frame $[H]$ with the robot's base orientation frame $[U]$, two subsequent rotations must be performed, see Figure 3.7. The rotation matrix ${}^U\mathbf{R}_H$, which can be used to express $[H]$ with respect to $[U]$ is obtained by

$$ {}^U\mathbf{R}_H = {}^U\mathbf{R}_* {}^*\mathbf{R}_H = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix}. \tag{3.2} $$

where $*$ denotes the frame between the two rotations as schematically depicted in Figure 3.7.



(a) Vive's coordinate frame.



(b) Correct alignment of the robot coordinate frame and Vive coordinate frame.

Figure 3.8: Illustration of the frames U and H as implemented on the real setup.

While calibrating the system, frame H is set accordingly to the pose of the HMD, as illustrated in Figure 3.8a. Another illustration of the teleoperation setup is displayed in Figure 3.8b, this figure shows the frames relevant for the teleoperation setup. In order to make sure that the orientation frame $[H]$ points in the same directions as $[U]$, it is important that the orientation of the head mounted display (HMD) is properly oriented. To this end, during calibration, the HMD must be placed such that the orientation of the coordinate frame of Vive $[H]$, is the same as the orientation of frame H in Figure 3.8b. After calibration, the HMD can be moved to a more convenient spot, but must always be visible to the two basestations. Using the methodology described here, the VR controller's pose is expressed with respect to the origin $\mathbf{o}_H$, and orientation frame $[U]$, by (3.3) and (3.4), respectively.

$$^{U}\mathbf{R}_V = {}^{U}\mathbf{R}_H{}^{H}\mathbf{R}_V \tag{3.3}$$

$$^{H[U]}\mathbf{o}_V = {}^{U}\mathbf{R}_H{}^{H}\mathbf{o}_V \tag{3.4}$$

Teleoperation Controls

Before we continue with the motion mapping between the leader device pose V and the robot's end-effector pose S, a description of how the operator commands the controller buttons is given. In each control step t_k, the so-called *button events* are retrieved from SteamVR. A button event is a signal that is retrieved by the controller every time a controller button on the leader device is pressed or released. These button event signals are used to control the suction gripper, but also to generate the reference trajectory for the robot's end-effector. The controller contains five buttons, see Figure 3.9, of which four are used to command the UR10 and suction gripper. These are the menu button (❶), touchpad (❷), trigger (❹) and grip button (❺), which are all used to either control the end-effector motion, or the I/O controls of the suction gripper. The grip button (❺) is implemented as a safety clutch to limit the risk of unexpected robot behavior that could damage the robot or its environment. For example, if the operator would drop the controller device. While this button is pressed, control commands can be given to the robot, if not, the controller button commands will not be communicated to the robot. The vacuum gripper is controlled using the trigger button (❹). By pressing the trigger, and an I/O signal is sent to the robot's control box, that activates the vacuum. This allows the robot to grasp onto objects. When the trigger is released, a I/O signal is sent to the robot's control box that deactivates the vacuum, and activates the blow-off function, causing the suction gripper to release the object. To stop both the vacuum and blow-off, the menu button (❶) can be pressed. Finally, the touchpad (❷) is implemented as a teleoperation clutch, which decides if the robot's end-effector pose must follow the leader device pose. How this exactly works is detailed in the remainder of this section.

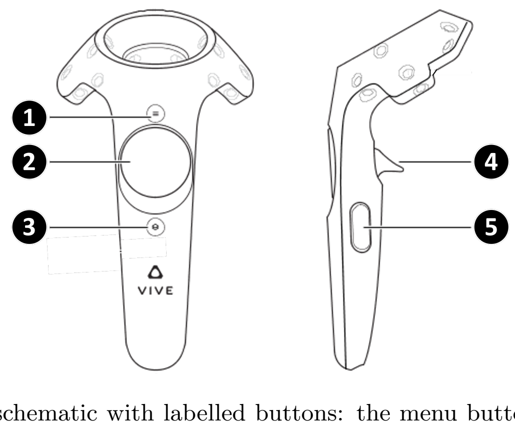


Figure 3.9: Vive controller schematic with labelled buttons: the menu button (1), touchpad (2), system button (3), trigger (4), and grip button (5).

Motion Mapping

As said before, the reference trajectory of the end-effector pose is generated using pose measurements of the leader device, the gripper button (❺), and the touchpad (❷). The desired end-effector pose, or reference input pose, with respect to the controller's inertial frame U, at time t is given by

$$ {}^{U}\mathbf{H}^{\text{ref}}_{S(t)} = \begin{bmatrix} {}^{U}\mathbf{R}^{\text{ref}}_{S(t)} & {}^{U}\mathbf{o}^{\text{ref}}_{S(t)} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{3.5} $$

where ${}^{U}\mathbf{R}^{\text{ref}}_{S(t)}$ and $\mathbf{o}^{\text{ref}}_{S(t)}$ denote the desired end-effector orientation and position, respectively. Similarly, the actual end-effector pose at time instance t, is denoted as ${}^{U}\mathbf{H}_{S(t)}$. At each time instance t, when ❺ and ❷ are not pressed at the same time, the desired end-effector pose at time instance t is given by ${}^{U}\mathbf{H}^{\text{ref}}_{S(t)} = {}^{U}\mathbf{H}_{S(t)}$. Hence, the reference input pose does not change. When ❺ and ❷ are both pressed, the desired end-effector pose at time instance t is computed using pose measurements of the leader device. In this case, the reference input position and orientation are computed by

$$ {}^{U}\mathbf{o}^{\text{ref}}_{S(t)} = {}^{U}\mathbf{o}_{S(t)} + \alpha \left({}^{H[U]}\mathbf{o}_{V(t)} - {}^{H[U]}\mathbf{o}_{V(t_p)} \right) \tag{3.6} $$

and

$$ {}^{U}\mathbf{R}^{\text{ref}}_{S(t)} = {}^{U}\mathbf{R}_{S(t_p)} \left({}^{H[U]}\mathbf{R}^{-1}_{V(t_p)} \, {}^{H[U]}\mathbf{R}_{V(t)} \right), \tag{3.7} $$

where t_p denotes the time instance at the first control loop iteration, since both the gripper and touchpad buttons are pressed. α denotes a positional reference scaling factor, and ${}^{H[U]}\mathbf{R}^{-1}_{V(t_p)}$ denotes the inverse orientation matrix of the leader device at time instance t_p. Note that the inverse of a rotation matrix is its transpose. Since the robot arm is relatively big compared to the human arm, intuitivity and effectivity might be improved by applying positional scaling, denoted by a scaling factor α, between the translational motion of the leader and follower device. Through (3.6), the current position of the leader device, relative to its position at t_p, is multiplied by a scalar α, and added to the current end-effector position of the robot. The scaling value is determined upon implementation, by experimenting on the real teleoperation setup. Note that the scaling factor is only applied to translational motion, for the rotations no scaling is applied. Through (3.7), the pose of the leader device at t_p is subtracted from the current pose of the leader device, and added to the current pose of the robot's end-effector.

Controller Algorithm

The control input is computed by means of an algorithm implemented through C++ objects. In the remainder of this section, the algorithm that generates the reference pose for the QP controller, as well as the commands for the suction gripper, is presented. The algorithm follows the methodology as described in this section. To refer to the states (pressed or released) of each button in the algorithm, we use ⓚ$_{\text{pressed}}$ or ⓚ$_{\text{released}}$, where k denotes the controller button denoted by a number 1-5, as given by Figure 3.9. Additionally, to store the device poses at t_p, a boolean (hereafter referred to as bool) is implemented in the control algorithm, which we refer to as `setRefPose`. This bool is initialized *true* at startup and used to store the end-effector and leader device poses at t_p. Every time ❺ and/or ❷ are released, the bool is set *true*, such that the next time these buttons are pressed the correct end-effector and leader device poses are stored.

For the teleoperation reference input controller algorithm, see Algorithm 1. The desired end-effector pose, computed by the algorithm, is used as reference input for the QP controller, which is detailed in Section 3.3.

Algorithm 1 Teleoperation reference input controller algorithm.

```
1:  for Each control loop iteration $t$ do
2:      Input: ${}^{H}\mathbf{H}_{V(t)}$ from SteamVR
3:      Input: ${}^{U}\mathbf{H}_{S(t)}$ from mc_rtc
4:      if ❺$_{\text{pressed}}$ then
5:          if ❷$_{\text{pressed}}$ then
6:              ${}^{U}\mathbf{R}_{V(t)} = {}^{U}\mathbf{R}_{H}{}^{H}\mathbf{R}_{V(t)}$
7:              ${}^{H[U]}\mathbf{o}_{V(t)} = {}^{U}\mathbf{R}_{H}{}^{H}\mathbf{o}_{V(t)}$
8:              if setRefPose then
9:                  $t_p = t$
10:                 ${}^{H[U]}\mathbf{R}_{V(t_p)} = {}^{H[U]}\mathbf{R}_{V(t)}$
11:                 ${}^{H[U]}\mathbf{o}_{V(t_p)} = {}^{H[U]}\mathbf{o}_{V(t)}$
12:                 ${}^{U}\mathbf{R}_{S(t_p)} = {}^{U}\mathbf{R}_{S(t)}$
13:                 ${}^{U}\mathbf{o}_{S(t_p)} = {}^{U}\mathbf{o}_{S(t)}$
14:                 setRefPose = $\textit{false}$
15:             end if
16:             Output: ${}^{U}\mathbf{o}^{\text{ref}}_{S(t)} = {}^{U}\mathbf{o}_{S(t)} + \alpha\left({}^{H[U]}\mathbf{o}_{V(t)} - {}^{H[U]}\mathbf{o}_{V(t_p)}\right)$
17:             Output: ${}^{U}\mathbf{R}^{\text{ref}}_{S(t)} = {}^{U}\mathbf{R}_{S(t_p)}\left({}^{H[U]}\mathbf{R}^{-1}_{V(t_p)}\,{}^{H[U]}\mathbf{R}_{V(t)}\right)$
18:         else if ❷$_{\text{released}}$ then
19:             Output: ${}^{U}\mathbf{H}^{\text{ref}}_{S(t)} = {}^{U}\mathbf{H}_{S(t)}$
20:             setRefPose = $\textit{true}$
21:         end if
22:         if ❹$_{\text{pressed}}$ then
23:             Output: Stop blow-off and start vacuum
24:         else if ❹$_{\text{released}}$ then
25:             Output: Start blow-off and stop vacuum
26:         else if ❶$_{\text{pressed}}$ then
27:             Output: Stop blow-off and stop vacuum
28:         end if
29:     else if ❺$_{\text{released}}$ then
30:         Output: ${}^{U}\mathbf{H}^{\text{ref}}_{S(t)} = {}^{U}\mathbf{H}_{S(t)}$
31:         setRefPose = $\textit{true}$
32:     end if
33: end for
```

3.2.3 Finite-State Machine Implementation

In this work, the Finite-State Machine (FSM) controller implementation provided by the mc_rtc framework was used. For an introduction to FSMs, please refer to Section 2.4. Through mc_rtc, by inheriting from a class of the framework, new modes can be programmed in C++. We use an FSM controller to build the appropriate order of control tasks in various phases of manipulation. For each task, a mode is build, which could contain one or multiple waypoints to guide the robot's motion, without the need to specify any explicit trajectory. We also programmed modes that activate the teleoperation framework, or repeat a previously demonstrated toss. Moreover, the FSM controller is appropriate to build variations of modes. In this case, for example, we made one teleoperation mode where the control inputs are stored such that these can be repeated later, and another teleoperation mode which does not store control inputs. The FSM controller in this work is defined by the modes listed below, whose names already imply the mode's functionality.

- *BoxToStart*, activates the conveyor for a given time, guiding the box into the corner.
- *MoveToStart*, guides the robot along specified waypoints that moves the robot into a neutral

configuration, or start position.

- *GrabBox*, guides the robot along waypoints, reaching for the specified box (Box 3, Box 4, or Box 5). When contact with the box is made, the vacuum gripper is activated, causing the gripper to hold onto the box. Subsequently, the box is lifted upwards to a final position. Additionally, an option (which is set by a boolean) to automatically grab a box at different positions and orientations is implemented. This way, experiments to analyze the sensitivity of pick poses on boxes can be executed very effectively (see Section 4.4 for details).
- *Teleoperation*, enables to control the robot and suction gripper using Vive's VR controller, allowing the operator to demonstrate tossing (or any other task). For details, see Section 3.2.2.
- *RecordTeleoperation*, is identical to the *Teleoperation* mode, but in addition it automatically logs the controller reference data at each time instance in a binary log, which can be replayed.
- *RepeatDemonstration*, can be appointed to any logged demonstration from the *RecordTeleoperation* mode, and replays the exact same reference control inputs, causing the robot to perform the same toss/task.
- *TerminateProcess*, stops the robot, terminates the process and shuts down the controller.

Each of the modes listed can be performed in any desired sequence, running in an infinite loop, or until the process is terminated by itself or an error. Moreover, each mode is implemented such that, with minor adaptions, OptiTrack recordings can be started and stopped with commands from the controller's modes.

3.2.4 GitLab

The software developed for the H2020 project I.AM. is available in one GitLab group. The teleoperation controller software developed during this project, is also made available to everyone involved with the I.AM. project, and can be found through the following link:

`https://gitlab.tue.nl/h2020-i-am-project/ur10-teleoperation-controller`

For information about how to download, install, and use the software, please refer to the GitLab project provided by the link above.

3.3 QP Control Approach

In Section 2.3, an introduction into QP control was given. In this Section, the QP control approach to control the robot's end-effector pose based on the reference position and orientation inputs as described in Section 3.2.2. First, the QP's minimization variables are presented. Then, the QP objectives are stated from which the the QP controller's cost function is derived. Then the constraints and how these are implemented is discussed. Finally, the final QP control problem is formulated.

The dynamics of a 6-DOF robot manipulator can be described using Lagrange dynamics. Neglecting friction and disturbance input, the dynamic model is described by Lagrange's equation:

$$\mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \mathbf{u}, \tag{3.8}$$

where $\mathbf{D}(\mathbf{q}) \in \mathbb{R}^{6\times 6}$ is the mass matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} \in \mathbb{R}^6$ is the vector of Coriolis and centrifugal forces, $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^6$ is the vector of gravity force, and $\mathbf{u} \in \mathbb{R}^6$ is the vector of inputs for joint control of the actuators.

To control the full-body motion of the UR10, the robot's end-effector pose S is controlled to a desired state S^{ref}. In the QP control approach, a constrained optimization problem is solved online during manipulation every Δt seconds, which in case of the UR10 is fixed at $\Delta t = 0.008$ seconds The outcome of the QP problem serves as input for the low-level control of each individual joint. In the remainder of this section, the different aspects of this QP problem are highlighted. First, the minimization variables are discussed, then the tasks that make up the cost function are described, and finally the constraints are stated.

3.3.1 Minimization Variables

The UR10 can be controlled by position or velocity control, which means that the output of this QP problem should be a set of joint-level positions or velocities. However, the joint positions and velocities depend explicitly on the joint accelerations. Moreover, by controlling the joint accelerations it is possible to command the desired inputs directly within one time step t_k, without generating too large motor torques, as would be the case if the QP is solved for the for position and velocity at each time step. Therefore, we choose the joint accelerations $\ddot{\mathbf{q}} \in \mathbb{R}^6$ as the minimization variables. The output of the QP problem is then integrated once, and used as input for low-level control of the individual joints for velocity control.

3.3.2 QP objectives

The quadratic cost function to minimize in the QP is defined in Equation (2.13), see Section 2.3. In the proposed teleoperation system, the QP controller objectives are described by two reference inputs: the desired end-effector position ${}^{U}\mathbf{o}_S^{\text{ref}}$ and orientation ${}^{U}\mathbf{R}_S^{\text{ref}}$, see Section 3.2.2. These QP objectives are described in the following two sections [48].

Position Task

The position task is used to control the robot's end-effector position ${}^{U}\mathbf{o}_S$, towards a desired end-effector position ${}^{U}\mathbf{o}_S^d$. To fit the QP problem discussed in Section 2.3, the cost function for the position task is formulated in quadratic form. This cost function, denoted by $\boldsymbol{E}^{\text{pos}}$, is given by

$$\boldsymbol{E}^{\text{pos}} = \frac{1}{2}\left\|{}^{U}\ddot{\mathbf{o}}_S - {}^{U}\ddot{\mathbf{o}}_S^d\right\|^2. \tag{3.9}$$

Here, ${}^{U}\ddot{\mathbf{o}}_S^d$ is given by [66]

$${}^{U}\ddot{\mathbf{o}}_S^d = {}^{U}\ddot{\mathbf{o}}_S^{\text{ref}} - k_p({}^{U}\mathbf{o}_S - {}^{U}\mathbf{o}_S^{\text{ref}}) - k_d({}^{U}\dot{\mathbf{o}}_S - {}^{U}\dot{\mathbf{o}}_S^{\text{ref}}) \tag{3.10}$$

where ${}^{U}\mathbf{o}_S^{\text{ref}}$ is the end-effector reference position as computed by the methodology discussed in Section 3.2.2. Moreover, $k_p, k_d \in \mathbb{R}$ denote scalar proportional and derivative gains, respectively. The damping k_d is determined by assuming a critically damped harmonic oscillator, hence $k_d = 2\sqrt{k_p}$. For desired contacts, such as parcels that are to be tossed by the system, the force exerted on an object is determined by a coupling between the chosen stiffness for the error between the actual end-effector pose, and the desired pose.

Since only the position and orientation of the leader device are measured, and the velocities and accelerations not, and it was sufficient to use only piecewise constant profiles of ${}^{U}\mathbf{o}_S^{\text{ref}}$, the reference velocities and accelerations have zero-values: ${}^{U}\dot{\mathbf{o}}_S^{\text{ref}} = 0$ and ${}^{U}\ddot{\mathbf{o}}_S^{\text{ref}} = 0$. Hence, the desired end-effector acceleration ${}^{U}\ddot{\mathbf{o}}_S^d$ for regulating the end-effector position around a constant value and steering it towards a target value, is calculated by

$${}^{U}\ddot{\mathbf{o}}_S^d = -k_p({}^{U}\mathbf{o}_S - {}^{U}\mathbf{o}_S^{\text{ref}}) - k_d{}^{U}\dot{\mathbf{o}}_S. \tag{3.11}$$

Orientation Task

The orientation task is used to control the orientation of the end-effector $[S]$, to the time varying desired orientation, denoted by ${}^{U}\mathbf{R}_{S(t)}^{\text{ref}}$. The computation of the reference orientation was discussed in Section 3.2.2. The quadratic cost function for the end-effector orientation is denoted by E^{ori}. This cost function is taken from [67], where its derivation, based on the PD controller on SO(3), can be found. Here, only the cost function itself is given, which is formulated as

$$\boldsymbol{E}^{\text{ori}} = \frac{1}{2}\|{}^{U}\dot{\boldsymbol{\omega}}_S - {}^{U}\dot{\boldsymbol{\omega}}_S^d\|^2 \tag{3.12}$$

where ${}^U\boldsymbol{\omega}_S \in \mathbb{R}^3$ and ${}^U\boldsymbol{\omega}_S^d \in \mathbb{R}^3$ denote the end-effector's current and desired angular velocity, respectively. ${}^U\dot{\boldsymbol{\omega}}_S^d$ is given by

$$ {}^U\dot{\boldsymbol{\omega}}_S^d = {}^U\dot{\boldsymbol{\omega}}_S^{\text{ref}} - {}^U\boldsymbol{\omega}_S \times \left({}^U\boldsymbol{\omega}_S^{\text{ref}} - {}^U\boldsymbol{\omega}_S\right) + k_d\left({}^U\boldsymbol{\omega}_S^{\text{ref}} - {}^U\boldsymbol{\omega}_S\right) + k_p {}^U\mathbf{R}_S\left(\log\left({}^U\mathbf{R}_S^{-1}{}^U\mathbf{R}_S^{\text{ref}}\right)\right)^\vee \tag{3.13}$$

where ${}^U\boldsymbol{\omega}_S^{\text{ref}} \in \mathbb{R}^3$ denotes the end-effector's reference angular velocity. The proportional and derivative gains are denoted by scalars $k_p, k_d \in \mathbb{R}$, and are determined by assuming a critically damped harmonic oscillator $k_d = 2\sqrt{k_p}$. The error between the orientation matrices ${}^U\mathbf{R}_S$ and ${}^U\mathbf{R}_S^{\text{ref}}$ is defined as the matrix logarithm of ${}^U\mathbf{R}_S^{\text{ref}}$ with respect to ${}^U\mathbf{R}_S$, as proposed in [67]. This error is computed by ${}^U\mathbf{R}_S^{-1}{}^U\mathbf{R}_S^{\text{ref}}$. The vee operator, denoted by $^\vee$, performs the inverse transformation, from a skew-symmetric matrix in $\mathfrak{so}(3)$ to a vector in $\mathbb{R}^3$, such that

$$ W^\vee = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}^\vee := \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3. \tag{3.14}$$

Again, since only the position and orientation of the leader device are measured, and the velocities and accelerations not, and it is sufficient to use only piecewise constant profiles of ${}^U\mathbf{R}_S^{\text{ref}}$, the reference angular velocities and angular accelerations have zero-values: ${}^U\boldsymbol{\omega}_S^{\text{ref}} = 0$ and ${}^U\dot{\boldsymbol{\omega}}_S^{\text{ref}} = 0$. Hence, the desired end-effector acceleration ${}^U\dot{\boldsymbol{\omega}}_S^d$ for regulating the end-effector position around a constant value and steering it towards a target value, is calculated by

$$ {}^U\dot{\boldsymbol{\omega}}_S^d = -k_d {}^U\boldsymbol{\omega}_S + k_p {}^U\mathbf{R}_S\left(\log\left({}^U\mathbf{R}_S^{-1}{}^U\mathbf{R}_S^{\text{ref}}\right)\right)^\vee . \tag{3.15}$$

Task-space to joint level conversion

To successfully control the end-effector, the task-space pose objective of the end-effector, controlled by the leader device pose, needs to be converted to joint-level objectives. The end-effector pose $S(\mathbf{q})$ can be determined from the joint positions $\mathbf{q}_i$ with $i \in \{1, \ldots, 6\}$, using forward kinematics. The body frame velocity and acceleration of S can be expressed using

$$ {}^U\mathbf{v}_S = \begin{bmatrix} {}^U\boldsymbol{v}_S \\ {}^U\boldsymbol{\omega}_S \end{bmatrix} = {}^U\mathbf{J}_S(\mathbf{q})\dot{\mathbf{q}} \tag{3.16}$$

and

$$ {}^U\dot{\mathbf{v}}_S = \begin{bmatrix} {}^U\dot{\boldsymbol{v}}_S \\ {}^U\dot{\boldsymbol{\omega}}_S \end{bmatrix} = {}^U\mathbf{J}_S(\mathbf{q})\ddot{\mathbf{q}} + {}^U\dot{\mathbf{J}}_s(\mathbf{q})\dot{\mathbf{q}}, \tag{3.17}$$

where the Jacobian matrix ${}^U\mathbf{J}_S$ relates the velocity of the body frame with respect to the inertial frame U.

3.3.3 Constraints

During human-operated manipulation, for example when gripping objects, the robot manipulator will interact with its environment. This raises the question of how to deal with interaction between the robot and its environment. If the operator controls the robot motion such that the desired trajectory of the robot goes through physical boundaries of the robot's environment, it must be made sure that the robot cannot damage its environment, and vice versa. In the first place, this is solved through collision constraints in the QP controller. In this section, it is presented which constraints are applied to the QP problem, and how this is done.

Joint limit constraints

The first constraints set is a result of the limits in the robot hardware, such as joint position, velocity and acceleration limits [68]. For each joint $\mathbf{q}_i$ with $i \in \{1, \ldots, 6\}$, these can be described by:

$$\underline{\mathbf{q}}_i \leq \mathbf{q}_i(t) \leq \overline{\mathbf{q}}_i, \tag{3.18}$$

$$\underline{\dot{\mathbf{q}}}_i \leq \dot{\mathbf{q}}_i(t) \leq \overline{\dot{\mathbf{q}}}_i, \tag{3.19}$$

$$\underline{\ddot{\mathbf{q}}}_i \leq \ddot{\mathbf{q}}_i(t) \leq \overline{\ddot{\mathbf{q}}}_i, \tag{3.20}$$

for any time t, with lower joint bounds for the joint position, velocity, and acceleration given by $\underline{\mathbf{q}}_i, \underline{\dot{\mathbf{q}}}_i$, and $\underline{\ddot{\mathbf{q}}}_i$ respectively, and upper bounds given by $\overline{\mathbf{q}}_i, \overline{\dot{\mathbf{q}}}_i, \overline{\ddot{\mathbf{q}}}_i$.

However, constraints (3.18) and (3.19) cannot be enforced directly at time t using the optimization variables $\ddot{\mathbf{q}}_i$, as there is no direct control over $\mathbf{q}_i$ and $\dot{\mathbf{q}}_i$. To overcome this issue, these constraints are discretized such that the predicted joint/link position and velocity after a single time step can be constrained [69]. To this end, a constant time step δt can be selected [70]. Here, this time step δt is set equal to the control time step of the controller Δt. The discretization is done using the forward Euler method to estimate the velocity at time $t + \delta t$ through

$$\dot{\mathbf{q}}_i(t+\delta t) = \ddot{\mathbf{q}}_i(t)\delta t + \dot{\mathbf{q}}_i(t), \tag{3.21}$$

for any time t. The joint positions are estimated using (3.21) and the backward Euler method (3.22), to obtain (3.23).

$$\mathbf{q}_i(t+\delta t) = \dot{\mathbf{q}}_i(t+\delta t)\delta t + \mathbf{q}_i(t) \tag{3.22}$$

$$\mathbf{q}_i(t+\delta t) = \frac{1}{2}\ddot{\mathbf{q}}_i(t)\delta t^2 + \dot{\mathbf{q}}_i(t)\delta t + \mathbf{q}_i(t) \tag{3.23}$$

The joint accelerations are chosen such that the joint position limits (3.18) and velocity limits (3.19) will not be violated at time $t + \delta t$. This leads to Equations (3.24) and (3.25), respectively.

$$\underline{\mathbf{q}}_i \leq \mathbf{q}_i(t+\delta t) \leq \overline{\mathbf{q}}_i \tag{3.24}$$

$$\underline{\dot{\mathbf{q}}}_i \leq \dot{\mathbf{q}}_i(t+\delta t) \leq \overline{\dot{\mathbf{q}}}_i \tag{3.25}$$

Finally, to gain control over the constraints, using (3.23) the joint position and velocity constraints are rewritten in terms of $\ddot{\mathbf{q}}_i$, see (3.26) and (3.27), respectively.

$$\underline{\mathbf{q}}_i \leq \frac{1}{2}\ddot{\mathbf{q}}_i(t)\delta t^2 + \dot{\mathbf{q}}_i(t)\delta t + \mathbf{q}_i(t) \leq \overline{\mathbf{q}}_i \tag{3.26}$$

$$\underline{\dot{\mathbf{q}}}_i \leq \ddot{\mathbf{q}}_i(t)\delta t + \dot{\mathbf{q}}_i(t) \leq \overline{\dot{\mathbf{q}}}_i \tag{3.27}$$

Through this discretization, the constraints are transformed to joint acceleration constraints that can directly be enforced using the optimization variables $\ddot{\mathbf{q}}_i$ with $i \in \{1, \ldots, 6\}$, and therefore can be added to the QP control formulation [69] [70].

Collision avoidance

The collision avoidance constraints are introduced to make sure the robot does not harm/damage itself, nor its environment. The collision avoidance in this work is applied by the methodology of [48]. This method allows for collision avoidance without the need for pre-computation or prediction of trajectories, and acts reactive to the motions that are present in the robot scene. The collision avoidance constraints are imposed as a set of a velocity-damper-based constraints. Consider two bodies a and b in the robot scene which require a constraint to avoid collision. The distance $\mathbf{d_{a,b}}$ between these bodies is

$$\mathbf{d_{a,b}} = \sigma||\mathbf{p}_a - \mathbf{p}_b||, \tag{3.28}$$

where $\mathbf{p}_a$ and ${}^U\mathbf{p}_b$ are the two *witness points*, and $\sigma = +1$ if there is no collision and $\sigma = -1$ if there is collision (in this case d is an inter-penetration distance). This distance is computed in the software using an implementation of the Gilbert, Johnson and Keerthi (GJK) algorithm [71]. At every configuration of the two convex bodies a and b, the GJK algorithm computes a witness point for each of the respective surfaces. These *witness points* are defined such that the distance between them is equal to the (minimal) distance between the two convex bodies. As the configuration of the bodies changes over time, these witness points move along the surfaces. The positive distance constraints are applied using these two moving points, thereby guaranteeing the satisfaction of the collision-avoidance constraint between the two convex bodies a and b, as they move over time. A basic velocity damper behavior is obtained through the following inequality

$$\dot{\mathbf{d}}_{\mathbf{a},\mathbf{b}} \geq -\xi \frac{\mathbf{d}_{\mathbf{a},\mathbf{b}} - \delta_s}{\delta_i - \delta_s}, \tag{3.29}$$

where the velocity damping coefficient, the influence distance between links below that the constraint starts to act, and the security distance between links that the constraint ensures will never be reached, are denoted by parameters ξ, δ_i, and δ_s, respectively. Inequality (3.29) can be used to prevent both self-collisions of a robot as collisions between a robot and its fixed environment. For a schematic overview of the collision avoidance method, see Figure 3.10.

However, the collision constraint as shown by inequality (3.29), cannot be directly enforced using the optimization variables $\ddot{\mathbf{q}}_i$ with $i \in \{1, \dots, 6\}$. Using the same method as used for the joint limit constraints, the collision constraints is rewritten to a QP-compatible version of 3.29, stated in (3.30).

$$\ddot{\mathbf{d}}_{\mathbf{a},\mathbf{b}} \geq \frac{1}{\delta t}\left(-\xi \frac{\mathbf{d}_{\mathbf{a},\mathbf{b}} - \delta_s}{\delta_i - \delta_s} - \dot{\mathbf{d}}_{\mathbf{a},\mathbf{b}}\right) \tag{3.30}$$

Denoting the unit vector $\mathbf{w}$ between $\mathbf{p}_a$ and $\mathbf{p}_b$ as $\mathbf{w} = (\mathbf{p}_a - \mathbf{p}_b)/d$, the derivative $\dot{\mathbf{d}}_{\mathbf{a},\mathbf{b}}$ is obtained by

$$\dot{\mathbf{d}}_{\mathbf{a},\mathbf{b}} = (\dot{\mathbf{o}}_a - \dot{\mathbf{o}}_b)^T \mathbf{w}. \tag{3.31}$$

Denoting $\mathbf{J}^{\text{pos}}_{\mathbf{o}_a}$ and $\mathbf{J}^{\text{pos}}_{\mathbf{o}_b}$ respectively, the linear (translational) Jacobians at $\mathbf{p}_a$ and $\mathbf{p}_b$, $\ddot{\mathbf{d}}_{\mathbf{a},\mathbf{b}}$ can be expressed as

$$\ddot{\mathbf{d}}_{\mathbf{a},\mathbf{b}} = \mathbf{w}^T\left(\mathbf{J}^{\text{pos}}_{\mathbf{o}_a}\ddot{\mathbf{q}}_a + \dot{\mathbf{J}}^{\text{pos}}_{\mathbf{o}_a}\dot{\mathbf{q}}_a - \mathbf{J}^{\text{pos}}_{\mathbf{o}_b}\ddot{\mathbf{q}}_b - \dot{\mathbf{J}}^{\text{pos}}_{\mathbf{o}_b}\dot{\mathbf{q}}_b\right) + \dot{\mathbf{w}}^T(\dot{\mathbf{o}}_a - \dot{\mathbf{o}}_b). \tag{3.32}$$

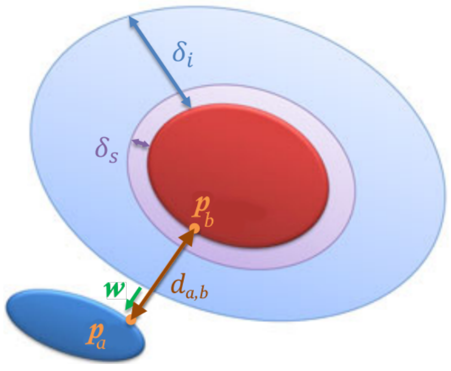


Figure 3.10: Illustration of the collision avoidance method where the blue (a) and red body (b) must avoid colliding [48]. The thin purple area wrapping body b represents the security zone within interaction with other bodies is considered as collision, the larger light blue zone represents the influence distance from which the constraint is activated and starts influencing the motions of the bodies. Using the distance $\mathbf{d}_{\mathbf{a},\mathbf{b}}$ between the witness points $\mathbf{o}_a$ and $\mathbf{o}_b$, it can be determined whether the constraint must be activated or collision has occurred. $\mathbf{w}$ is the unit vector between $\mathbf{o}_a$ and $\mathbf{o}_b$.

As can be seen in (3.32), $\ddot{\mathbf{d}}_{\mathbf{a},\mathbf{b}}$ can be enforced directly in the minimization variables $\ddot{\mathbf{q}}$, allowing the constraints to be added to the QP control formulation.

An analysis has been made to investigate which possible (self-)collisions could occur to find out which constraints must be applied to the QP problem. In Table 3.1, each link ℓ_i with $i \in \{1, \dots, 7\}$ of the UR10 (see robot description in Section 3.1.1), together with the testbench and conveyor belt bodies (respectively b_1 and b_2) are listed to compare which links could possibly collide. Each of the links (rows) that could collide with another link (columns) is marked with an "×". Since a reduced number of constraints leads to less computational effort, the number of constraints is minimized to the required amount for safe operation. By only applying constraints between links that are marked with an "×*" (see Table 3.1), self-collisions in the whole robot can be avoided, while avoiding the need for constraints between each link. This results in self-collision avoidance based on only six constraints, which are all applied to the QP problem.

Table 3.1: Possible (self-)collisions of the UR10.

	Shoulder	**Upper arm**	**Forearm**	**Wrist 1**	**Wrist 2**	**Wrist 3**	**Gripper**
ℓ_1: Shoulder							
ℓ_2: Upper arm							
ℓ_3: Forearm	×*						
ℓ_4: Wrist 1		×					
ℓ_5: Wrist 2	×*	×	×				
ℓ_6: Wrist 3	×*	×	×				
ℓ_7: Gripper	×*	×*	×*				
b_1: Testbench		×*	×*	×*	×*	×*	×*
b_2: Conveyor		×	×*	×*	×*	×*	×*

3.3.4 Final QP Formulation

Combining the cost function formulated in (2.13) and the constraints given by (3.26), (3.27), and (3.20), the final QP problem that is implemented is stated below.

$$\min_{\ddot{\mathbf{q}}} \quad c_{\mathbf{q},\dot{\mathbf{q}}}(\ddot{\mathbf{q}}) = w^{\text{pos}} \boldsymbol{E}^{\text{pos}} + w^{\text{ori}} \boldsymbol{E}^{\text{ori}} \tag{3.33a}$$

$$\text{s.t.} \quad \mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \mathbf{u} \tag{3.33b}$$

$$\ddot{\mathbf{q}}_i \delta t + \dot{\mathbf{q}}_i \geq \underline{\dot{\mathbf{q}}}_i \tag{3.33c}$$

$$\ddot{\mathbf{q}}_i \delta t + \dot{\mathbf{q}}_i \leq \overline{\dot{\mathbf{q}}}_i \tag{3.33d}$$

$$\frac{1}{2}\ddot{\mathbf{q}}_i \delta t^2 + \dot{\mathbf{q}}_i \delta t + \mathbf{q}_i \geq \underline{\mathbf{q}}_i \tag{3.33e}$$

$$\frac{1}{2}\ddot{\mathbf{q}}_i \delta t^2 + \dot{\mathbf{q}}_i \delta t + \mathbf{q}_i \leq \overline{\mathbf{q}}_i \tag{3.33f}$$

$$\ddot{\mathbf{q}}_i \geq \underline{\ddot{\mathbf{q}}}_i \tag{3.33g}$$

$$\ddot{\mathbf{q}}_i \leq \overline{\ddot{\mathbf{q}}}_i \tag{3.33h}$$

$$\ddot{\mathbf{d}}_{\mathbf{a},\mathbf{b}} \geq \frac{1}{\delta t}\left(-\xi \frac{\mathbf{d}_{\mathbf{a},\mathbf{b}} - \delta_s}{\delta_i - \delta_s} - \dot{\mathbf{d}}_{\mathbf{a},\mathbf{b}}\right) \tag{3.33i}$$

In the QP formulation above, constraints (3.33c) to (3.33h) are active for each joint $\mathbf{q}_{\mathbf{1},\dots,\mathbf{6}}$. To avoid collisions, (3.33i) is active on the sets of bodies market by an × in Table 3.1. The QP task weights w^{pos} and w^{ori} are set to be equal, and are tuned upon system implementation.

This QP problem (3.33) is solved online in mc_rtc at every control time-step. The resulting $\ddot{\mathbf{q}}$ is integrated to send a velocity reference for velocity control. Moreover, the system is controlled in open loop, to update the state of the system for the next time-step iteration, the calculated states $(\mathbf{q}, \dot{\mathbf{q}})$ are used.

Chapter 4

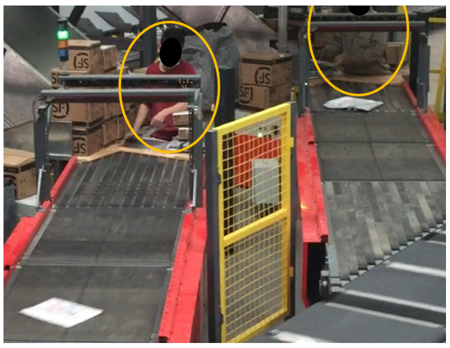
Validation

In this section, the functionality of the realized teleoperation system for tossing is experimentally validated. We also assess the applicability of robotic tossing in industry. To this end, we present a use case for robot tossing operations in logistic warehouses, see Section 4.1. Based on this use case, the experiments and requirements for tossing outcomes are established. Then, in Section 4.2, the general setup of the experiments at Vanderlande's Innovation lab at the TU/e campus is described. Subsequently, the repeatability of robotic tossing is assessed, which is presented in Section 4.3. In another experiment, an exploration is made to get an idea of how the position and orientation of the suction gripper with respect to the box affects the outcome of a toss. We refer to this experiment as the sensitivity study, which is presented in Section 4.4.

4.1 Use Case for Robotic Tossing

In this section, the use cases for robotic tossing in industry is presented, to give an idea of in which logistics processes tossing can be applied. Vanderlande identified two logistic processes in industry as potentially interesting for the usage of tossing operations to increase the throughput. To this end, the crossorter infeed [72] and traysorter infeed [73] processes were identified, see Figure 4.1a and 4.1b, respectively. Both these processes are part of larger logistics sortation systems where unsorted objects coming in unstructured bins, containers, or pallets, are added to a sortation system.

In the crossorter infeed process, human operators or Vanderlande's SIR system (see Figure 1.1, Chapter 1), place the objects on a 1.3 m wide conveyor, running at 1.5 m/s. From the conveyor, these objects are fed into the crossorter, which is a continuous train of carts moving at 2 - 2.5


(a) Scenario 1: crossorter infeed [72].

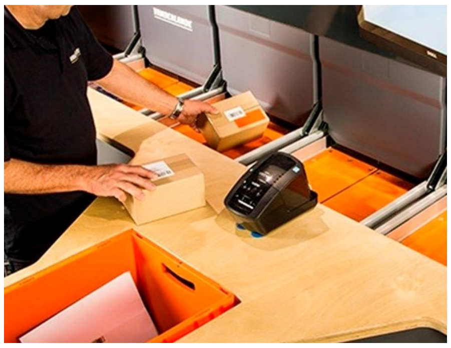

(b) Scenario 2: traysorter infeed [73].

Figure 4.1: Scenarios for applying autonomous robot tossing.

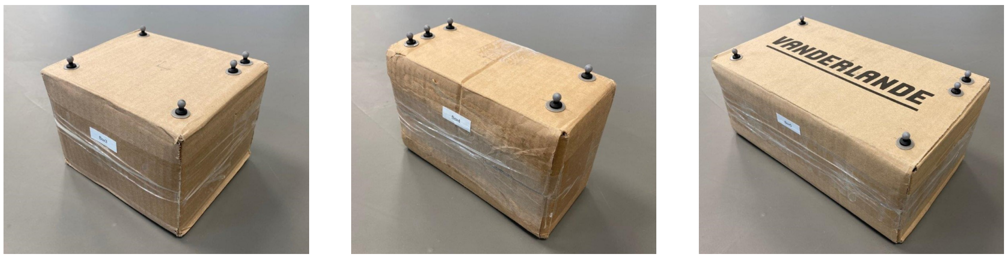
Table 4.1: Tossing scenario specifications.

Specifications:	**Scenario 1: Crossorter Infeed**	**Scenario 2: Traysorter Infeed**
Conveyor speed:	1.5 m/s	1.5 m/s
Target area size:	1300×2100 mm	400×400 mm
Target orientation:	Barcode not pointing down and short-edge leading, with an orientation of $30° \pm 5°$ with respect to the crossorter	Barcode pointing up, any orientation
Parcel weight:	< 8 kg (67% < 1 kg)	< 8 kg (67% < 1 kg)

m/s, each containing a conveyor belt perpendicular to the driving direction. While placing objects on the infeed conveyor, the object's barcode must be visible (i.e. not pointing down), and the parcel is placed such that its orientation matches the orientation of the crossorter belt within a maximum deviation of 30°. In the traysorter infeed process, picked items are (manually) placed in trays on a sorter, running at a speed of 1.5 m/s. These trays have dimensions of 400×400 mm, or 600×400 mm. While placing objects in a tray, any orientation is allowed, as long as the barcode is pointing up. An overview of the specifications and placement requirements of each process are given in Table 4.1.

In both processes, a wide variety of objects is handled: envelopes, bags, and boxes, in many combinations of size, shape and weight. A group of parcels that is identified by Vanderlande to increase throughput by tossing, is the so-called *smalls* category. Smalls parcels are typically envelopes, boxes, or plastic-wrapped items, up to 8 kg. A study of Vanderlande into smalls parcels at a parcel site found that the average parcel weight was 0.78 kg, with parcels weighing 0.4 kg being the most common. The dimensions of parcels were also investigated in this study, this revealed that parcel dimensions typically don't exceed 30 cm. For reference, a global survey with over 33 thousand respondents by Statista was reviewed, regarding the average weight of packages in cross-border deliveries worldwide over 2021 [74]. In correspondence with Vanderlande's study, this revealed that the most common parcel weighs in the range of 0.2-0.5 kg (29%), and that 67% of all parcels weigh less than 1 kg.

Unfortunately, we cannot assess the complete range of parcels for robotic tossing in this project. The behavior of a parcel during tossing and while landing differ immensely per parcel type. A carton box might shift and tumble, plastic bags, however, are not likely to roll or bounce back up. To limit the scope, this work focuses on filled carton boxes. To this end, three typical smalls parcels that are identified by Vanderlande, see Figure 4.2. These boxes are available in the Vanderlande Innovation lab at the TU/e campus, to which we refer as Box 3, Box 4, and Box 5. These carton boxes are a good representation of the parcels that are handled in the crossorter



(a) Box 3. (b) Box 4. (c) Box 5.

Figure 4.2: Three boxes provided by Vanderlande that are representative for the smalls parcels in industry. Each box is equipped with 6.4 mm reflective markers on its top surface to allow motion tracking with the OptiTrack mocap system

Table 4.2: Dimensions, mass and inertia properties of Box 3, Box 4 and Box 5.

Specifications:	**Unit**	**Box 3**	**Box 4**	**Box 5**
Mass:	kg	0.316	0.297	0.566
Dimensions ($\ell \times w \times h$):	mm	$165 \times 143 \times 117$	$198 \times 93 \times 129$	$272 \times 155 \times 114$
Inertia $\{I_{xx}, I_{yy}, I_{zz}\}$:	kg mm^2	$\{899, 1077, 1255\}$	$\{626, 1382, 1184\}$	$\{1746, 4103, 4623\}$

infeed and traysorter infeed processes of Vanderlande. Each box is uniformly filled with a hard foam block that goes by the name of *sculpture block foam SB30* [75]. By using this material, the boxes can be filled uniformly, and the center of mass of each box lays approximately in its geometrical center. In Table 4.2, the properties of each box can be found. Moreover, each box is equipped with five OptiTrack reflective markers, which allow for capturing real-time motion data of the boxes while being tossed.

4.2 General Setup

The experiments are carried out in the Vanderlande Innovation lab, at the TU/e campus. An overview of the setup in the lab, at the time of the experiments, is given in Figure 4.3. The teleoperation system's hardware setup is described in Section 3.1.3. For a complete description of Vanderlande's Innovation lab at the TU/e campus, please refer to Appendix A. During the experiments, the joint limit velocities and accelerations of the UR10 robot are set to its maximum limits, which can be found in Table 4.3. During the experiments, we use the OptiTrack motion capture system to track bodies in 3D space. By applying OptiTrack markers to the robot, conveyor, and boxes, the pose of each component is tracked. For details about the OptiTrack system, please refer to Appendix A.2. Additionally, for reference to the measurements during each experiment, an Intel RealSense D415 camera is used to record video footage. The OptiTrack recordings and RealSense recordings are automatically started and stopped using digital control outputs from the controller. On top of that, by default, controller inputs and outputs such as iteration time, encoder values, body poses, and joint commands are logged by the control software (see Section 3.1.4). Moreover, the control inputs to repeat tossing recordings are saved, such that tossing motions can be repeated any time later. For each experiment, all empirical data is saved and linked to the metadata of the experiment, then this data is added to the TU/e impact-aware robotics database[1], which is part of the I.AM. project.

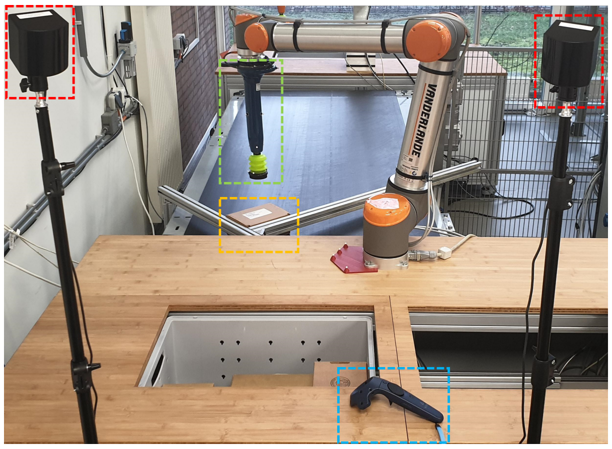
Summed up, the following (empirical) data is generated for each robot toss:

- Video recording of the complete experiment using a RealSense camera, on which the box and robot are clearly visible.
- OptiTrack motion capture data of the SR suction gripper, suction lip, and box.
- QP controller log, containing the following information: iteration time (starting from 0, increasing by the configured time step ($\Delta t = 0.008$)), encoder values of the robot (stored in the robot's reference joint order), orientation (as a quaternion) and position for the main robot (controller reference), joint commands for the robot (controller reference, stored in the robot's reference joint order).
- QP control input log to replay the robot tossing/task demonstration.

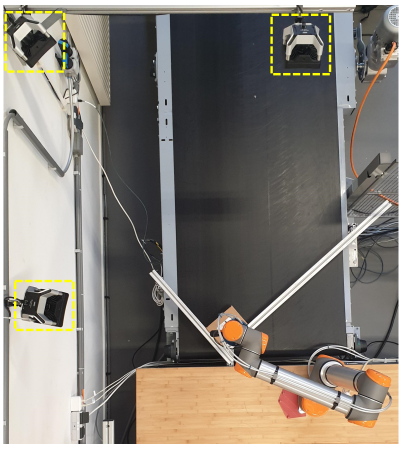
Table 4.3: Joint velocity and acceleration limits of Universal Robot's UR10 [49].

Joints:	**Base**	**Shoulder**	**Elbow**	**Wrist 1**	**Wrist 2**	**Wrist 3**
Joint velocity limit:	$131°/s$	$131°/s$	$131°/s$	$191°/s$	$191°/s$	$191°/s$

[1]See: `https://www.impact-aware-robotics-database.tue.nl/`



(a) Front view.



(b) Top view.

Figure 4.3: Overview of the teleoperation setup during the experiments. The front view (a) shows two Basestations, one VR controller, a box and the robot's end-effector are indicated by the red, blue, orange and green dotted lines, respectively. The top view (b) shows three of the six OptiTrack Cameras, indicated by the yellow dotted line.

Table 4.4: Final controller parameters of the teleoperation controller.

Controller stiffness	**QP task weight ratio**	**Position reference scaling**
$k_p = 45\ \text{Nm}^{-1}$	$w^{\text{pos}}/w^{\text{ori}} = 10/1$	$\alpha = 2$

To improve the teleoperation system for tossing operations, various controller parameters such as the controller's position and rotational stiffness, QP task weight, and position reference scaling can be tuned, see Section 3.3. By changing these parameter values, the teleoperation system can be tuned for better performance in terms of intuitiveness, and easier task demonstration. By means of experiments, these parameter values are broadly tuned. An experiment is defined to quantify reasonable parameter settings, such that the teleoperation system can effectively be used to demonstrate tossing operations. The final parameters resulting from the experiments are given in Table 4.4, which are used in the controller, see Section 3.3. For a description of the complete experiment, please refer to Appendix H.

4.3 Robot Tossing Repeatability

Through the teleoperation system presented in this work, human intuitivity can be exploited to perform robotic tossing operations, enabling to quickly generate motion trajectories for tossing to desired locations. In order to provide robotic solutions for automated tossing in industry, it is important to determine whether a tossing trajectory can be repeated by the robot hardware, thereby leading to similar tossing outcomes. If it can be proved that a tossing outcome can be repeated, the outcome of this toss becomes predictable. We define repeatability as the variation between the results of successive tossing demonstrations, under the same conditions of measurement (e.g., same robot, object, environment, location, measurement instrument, and procedure), in a short period of time [76]. The repeatability of a robot toss is reported in terms of standard deviation and variance of the final pose of the box, after a toss.

In this section, first, the goal of the experiments is presented, see Section 4.3.1. Then, in Section 4.3.2 the experimental setup is presented. Subsequently, the procedure to determine whether the robot hardware is capable of reproducing a tossing result is presented in Section 4.3.2. Finally, the results are presented and discussed in Section 4.3.3.

4.3.1 Goal of the Repeatability Study

We aim to show if a human demonstrated toss with a successful tossing result, can be repeatedly executed by the robot, leading to predictable tossing results. To this end, we perform repetitions of tossing motions using the robot hardware. By computing the deviation in the box' rest poses after each toss, we can analyze how the rest poses after each repetition distribute about the mean. This way, it can be demonstrated if repeated open-loop tossing is effective. The research question for this experiment is:

Can a human-demonstrated toss on a target surface, which lands on its bottom side and does not tumble after impact, repeatedly be performed by the robot hardware, and thereby lead to predictable tossing results?

4.3.2 Setup and Procedure of the Repeatability Study

In the experiments, a distinction is made between short and long range tossing, as well as between tossing on a stationary and a running conveyor belt. In the current setup, by default, the speed of the running conveyor is 0.4 m/s. Note that this speed does not correspond to the speed that is utilized in crossorter and traysorter processes (see Section 4.1). Due to time constraints, we could not perform the experiments for a conveyor speed of 1.5 m/s, as is utilized in the crossorter infeed and traysorter infeed processes, was not implemented. However, at a speed of 0.4 m/s, the difference between tossing on a running and idle conveyor can still be examined. The tossing repeatability for Box 5 is examined for four different situations: (1) a short range toss on a stationary conveyor belt, (2) a long range toss on a stationary conveyor belt, (3) a short range toss on a running conveyor belt, and (4) a long range toss on a running conveyor belt. In this context, a short range toss is defined as a toss that results in a box end pose that is inside the robot's kinematic range. A long range toss is defined to result outside the robot's kinematic range. To indicate this difference, Figure 4.4 displays two pictures of tossing trajectories of a toss where the box lands inside the robot's kinematic range, and one toss where the box lands outside the robot's kinematic range.

During each experiment, the same UR10 robot, at the same lab, is controlled using the same control inputs. Moreover, the measurements are done with the same measuring instrument (OptiTrack), under the same conditions, and the repetitions of demonstrations are done over a short period of time (< 30 seconds). Moreover, the repeatability of a demonstrated toss is examined using Box 5 (see Section 4.1). Box 5 has proved to be easily trackable using the OptiTrack system, and is the best representation for the *smalls* category parcels, in terms of dimensions and mass. However, since it has been tossed extensively in other projects, the edges and surfaces of Box 5 are rather rounded. Because of this, we expect more deviation in the rest-pose of Box 5 after a

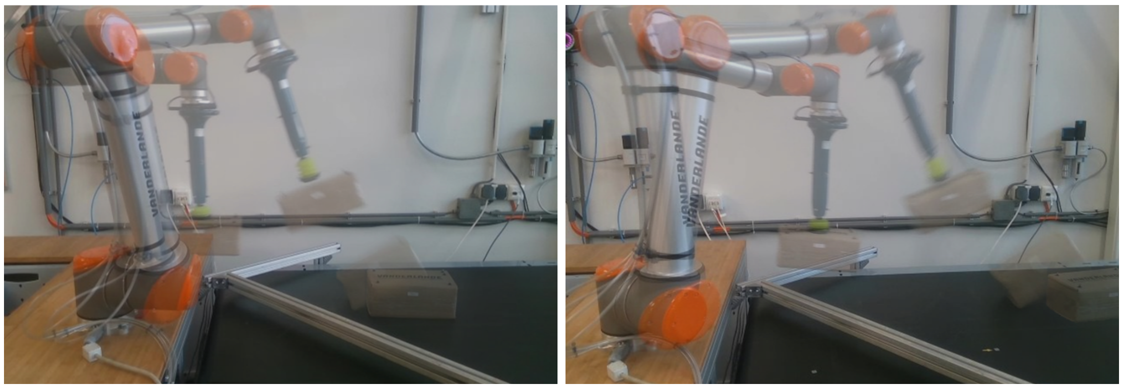

(a) Short range toss motion trajectory. (b) Long range toss motion trajectory.

Figure 4.4: Visualization of the tossing trajectories for the short and long range tossing motions.

toss, compared to tossing outcomes of a box with sharp edges. We chose to use Box 5 for the experiments, assuming that the results will represent the worst case scenario of predictability in tossing outcomes. Moreover, due to time constraints, there was no time to make new boxes, as this would also require ordering new materials and identifying its properties.

The process during the experiments is basically divided in two parts: first, the operator demonstrates a toss, after this, the robot repeats the same (human demonstrated) toss. Before each toss, the box is automatically guided back to a predefined position, picked up by the robot, and moved to a start position, each time in the same way. To carry out this process, the FSM controller implementation, as introduced in Section 3.2.3, is used. A sequence of pictures that shows the robot in motion during an experiment, is shown in Figure 4.5. Four FSM modes are implemented, Figure 4.6 depicts a schematic of the FSM modes and transitions. In this figure, the colors below each picture correspond with the colors of the modes as depicted in Figure 4.5.

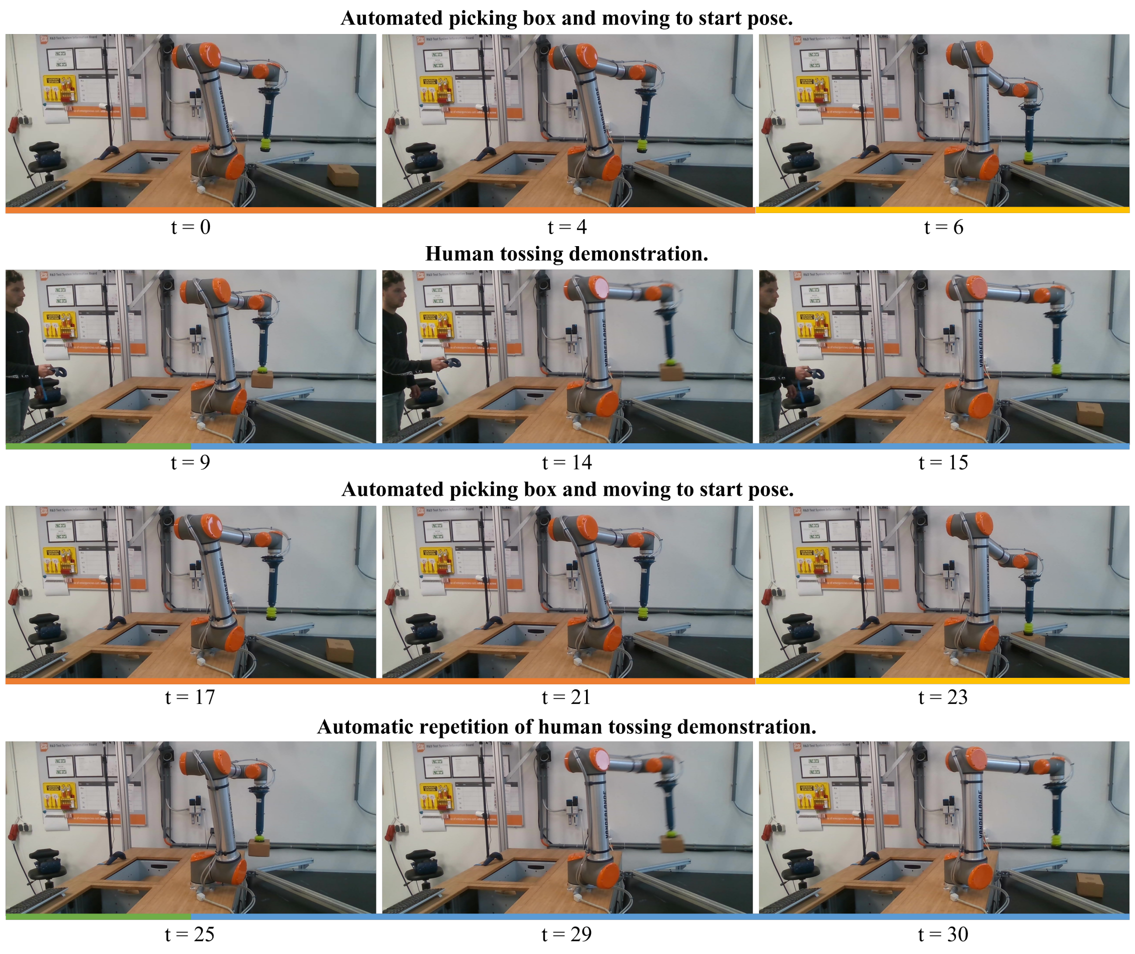

Automated picking box and moving to start pose.
t = 0
t = 4
t = 6
Human tossing demonstration.
t = 9
t = 14
t = 15
Automated picking box and moving to start pose.
t = 17
t = 21
t = 23
Automatic repetition of human tossing demonstration.
t = 25
t = 29
t = 30

Figure 4.5: FSM machine sequence with images at specified times, showing a human tossing demonstration and an automated repetition of the tossing demonstration. The colored line under the photos is used in correspondence to the colors of the FSM modes in Figure 4.6. During $t = [9, 15]$ s, a human tossing demonstration is performed. During $t = [25, 30]$ s, the inputs during the tossing demonstration are automatically repeated by the robot.

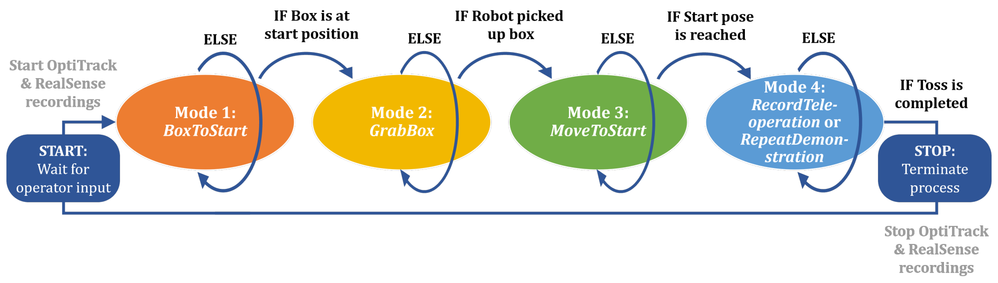

Start OptiTrack & RealSense recordings
START: Wait for operator input
ELSE
IF Box is at start position
Mode 1: BoxToStart
ELSE
IF Robot picked up box
Mode 2: GrabBox
ELSE
IF Start pose is reached
Mode 3: MoveToStart
ELSE
Mode 4: RecordTele-operation or RepeatDemon-stration
IF Toss is completed
STOP: Terminate process
Stop OptiTrack & RealSense recordings

Figure 4.6: Overview of the FSM modes that are implemented during the experiments. The controller loops through each mode until a requirement is met, at which point it moves on to the next mode.

When the operator activates the FSM controller, in the first mode (Mode 1) the conveyor is activated for four seconds from $t = [0, 4)$ s. This way, the box is guided into the positioning hook, ensuring that the robots consistently picks the box in the same position and orientation. Then, from $t = [4, 6)$ s during the second mode (Mode 2), the robot automatically picks the box with a predefined suction gripper pose. Subsequently, during $t = [6, 10)$ s (Mode 3), the robot lifts and moves the box into a start position. After this, in the teleoperation mode (Mode 4), during $t = [10, 15)$ s, the operator gains control over the robot motion by controlling the robot's end-effector motion using the teleoperation leader device. During this mode, the operator performs a tossing demonstration The control inputs during this mode are logged such that the motion can be repeated at a later stage. When an appropriate short and long range tossing demonstration are recorded, the process can be repeated, but this time the robot repeats the demonstrated toss automatically. To this end, modes 1, 2 and 3 are repeated during $t = [15, 23)$ s. Note that the process during $t = [17, 23]$ s is the same as during $t = [0, 6]$ s. Finally, over $t = [25, 30)$ s, the *RepeatDemonstration* mode (Mode 4) is active, during which the recorded control inputs of the tossing demonstration are replayed by the robot. The repeated tossing motion leads to the same tossing result as at $t = 15$. When the box has landed, the RealSense and OptiTrack recordings are stopped and stored. The OptiTrack recordings are used to determine the box' rest-pose after each toss. Note that Mode 4 is either a teleoperated tossing demonstration of the operator in real-time, or a repetition of the recorded tossing demonstration.

The first step is to generate an appropriate short and long tossing trajectory by human demonstration. The tossing demonstrations must involve a very stable toss on the conveyor, for which the box does not tumble. In this context, the tossing demonstration must lead to a tossing outcome where the box lands on its bottom without tumbling, and the orientation of the box' end-position is $< \|15°\|$ with respect to the conveyor. In addition, the hypothesis is that the outcome of a toss is more predictable if the box does not tumble after impact. To limit the chance of tumbling, the angle between the conveyor and the box' bottom surface will be $< \|30°\|$ at the moment of impact. After impact, this angle should remain $< \|30°\|$ until the box reached its final pose. Through these requirements, we assume repeatable tossing can be achieved. To confirm if a tossing demonstration complies with the requirements, we make video recordings of the box' motion. The final short- and long range tossing trajectories are shown in Figures 4.4a and 4.4b, respectively. A total of 30 tosses for both the short and long range toss are repeated. For the reasoning behind the choice for this sample size, please refer to Appendix I.2.

Consider Figure 4.7 for an illustration of a box after tossing. The result of a tossing demonstration, which we refer to as the box' rest-pose, is expressed as the position and orientation of B, with respect to the robot's base frame U. The rest-pose of a box is reached when the box stops moving with respect to the conveyor. For each toss, where the box lands on the same side, $({}^{U}\mathbf{o}_B)_{\mathrm{z}}$ is constant since the box end up at the same height every time. Therefore, to describe the difference between the box' rest-positions after tossing, we only use $({}^{U}\mathbf{o}_B)_{\mathrm{x}}$ and $({}^{U}\mathbf{o}_B)_{\mathrm{y}}$. Moreover, as can be seen in Figure 4.7, $[B]$ is only rotated over $\mathbf{z}_U$. Hence, the orientation of frame B with respect to $[U]$ can be given by

$$^{U}\mathbf{R}_B(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

where θ denotes the angle between $[B]$ and $[U]$. To this end, we express the rest-pose of a box after the i^{th} toss by

$$\mathbf{b}_i = \left[\left({}^{U}\mathbf{o}_B\right)_{\mathrm{x},i} \quad \left({}^{U}\mathbf{o}_B\right)_{\mathrm{y},i} \quad \theta_i \right]^T . \tag{4.2}$$

While the rest-pose of the box on a stationary conveyor is easily determined, for tossing on a running conveyor, we need to find a way to determine the rest pose of the (moving) box. To determine the rest-pose of a box in case of a running conveyor, we defined a condition: the rest pose of the box for toss $i = \{1, \ldots, 30\}$, is defined to be $\mathbf{b}_i(k_r)$, where k_r is the index of the OptiTrack measurement time t_o, at the first moment when the rate of change between the position of the box ${}^{U}\mathbf{o}_B$ and one of the conveyor frames ${}^{U}\mathbf{o}_{C_j}$ is below 1 mm, during 0.5 seconds. Mathematically, this condition is expressed by

$$\|{}^{U}\mathbf{o}_{C_j}(k) - {}^{U}\mathbf{o}_B(k)\| < 10^{-3} \text{ mm}, \quad \text{for} \quad k \in [k_r, k_r + 180], \tag{4.3}$$

where k denotes the time index of the OptiTrack measurements, given by $t_o = k/360$ for $k = 1, 2, \ldots$, and ${}^{U}\mathbf{o}_{C_j}$ with $j = \{1, \ldots, 6\}$ denotes a rigid body frame that is fixed on the moving conveyor.

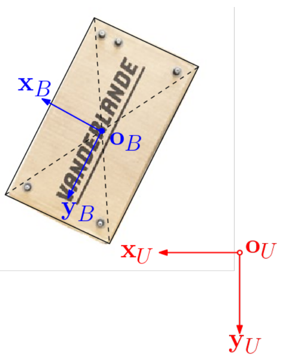


Figure 4.7: The pose of Box 5 expressed, given by B and the UR10 robot base frame U.

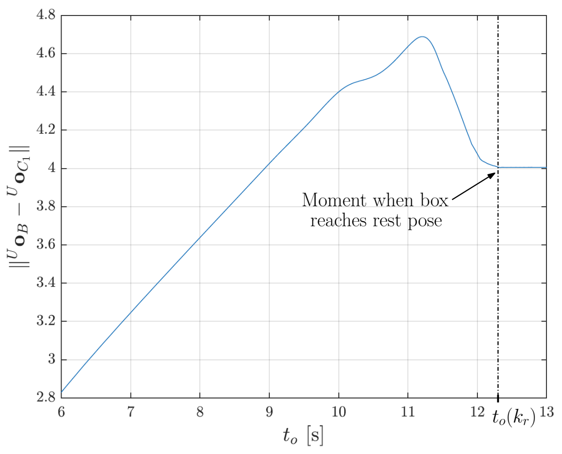


Figure 4.8: Example of 2-norm between measurements of ${}^{U}\mathbf{o}_B$ and ${}^{U}\mathbf{o}_{C_1}$ at each time instance t_o, to determine the box' rest-pose for tossing on a running conveyor.

The six conveyor rigid body frames are composed by six sets of five OptiTrack markers that are placed on the conveyor, in a way that always at least one conveyor frame C_j is visible. For more information about the OptiTrack system, please refer to Appendix A.2. While the conveyor is running, we can always measure the position ${}^{U}\mathbf{o}_{C_j}$ of at least one frame during the experiments. Using the OptiTrack measurements, the difference between the box' pose B and the pose of at least one conveyor frame C_j can be calculated at each time instance. Figure 4.8 shows the moment of rest $t(k_r)$ for one of the tossing experiments. In the post-processing scripts that are written in MATLAB, an algorithm automatically determines the rest-pose of the box after each toss. In the remainder of this section, the repeatability results of each experiment are presented and discussed.

4.3.3 Repeatability Results and Analysis

Repeatability is defined as the variation between the results of successive tossing demonstrations, under the same conditions of measurement (e.g., same robot, object, environment, location, measurement instrument, and procedure), in a short period of time [76]. We studied if the same box, given the same conditions, can be tossed to the same location, for 30 repetitions. We assume that the box rest poses follow a Gaussian distribution. The repeatability of a tossing demonstration is expressed in terms of the standard deviation and precision of the box' rest-pose parameters. We denote the standard deviation of the rest-pose parameters $\left({}^{U}\mathbf{o}_B\right)_{\mathrm{x}}$, $\left({}^{U}\mathbf{o}_B\right)_{\mathrm{y}}$, and θ, by σ_{x_b}, σ_{y_b}, and σ_{θ_b}, respectively. Since we compute the standard deviation using a limited number of samples, we report the standard deviation with a confidence interval of 95%. For more information about how we compute the standard deviation and confidence interval, please refer to Appendix I. We define the precision of the tossing outcomes by the difference between the minimum and maximum value of each box rest-pose parameter. For example, the precision of the box' rest pose in direction $\vec{x}_U$, is given by

$$\Delta\left({}^{U}\mathbf{o}_B\right)_{\mathbf{x}} = \left({}^{U}\mathbf{o}_B\right)_{\mathrm{x}}^{\max} - \left({}^{U}\mathbf{o}_B\right)_{\mathrm{x}}^{\min}. \tag{4.4}$$

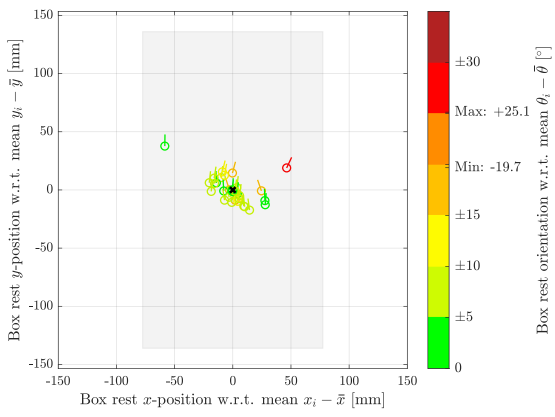
Similarly, the precision of the box' rest pose in direction $\vec{y}_U$, and the precision of the box' rest-pose orientation θ are given by $\Delta\left({}^{U}\mathbf{o}_B\right)_{\mathbf{y}}$ and $\Delta\boldsymbol{\theta}$, respectively.

Short Range Tossing Repeatability Results

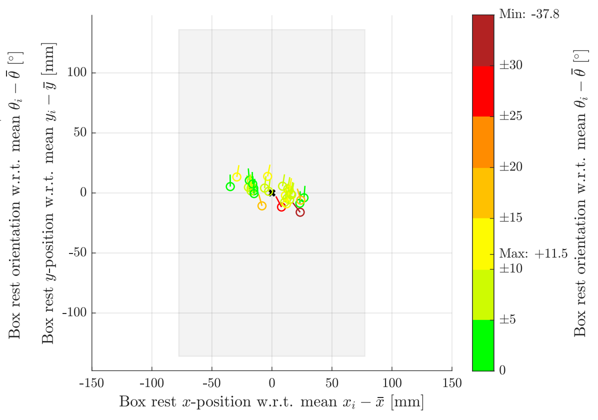
After repeating 30 short range tosses on both an idle and running conveyor, the repeatability of a short range toss for Box 5 is determined. The rest-poses after each toss $\mathbf{b}_i$ are displayed with respect to the mean rest-pose after tossing $\bar{\mathbf{b}}$, where position is expressed in millimeters, and orientation is expressed in degrees. In Figures 4.9a and 4.9b, the box' rest-poses after each toss for short range tossing are displayed for an idle and running conveyor, respectively. In Table 4.5, the repeatability results can be found. For the sake of completeness of the results, the mean rest-pose $\bar{\mathbf{b}}$ of the box with respect to the UR10 robot base coordinate frame U, is $\bar{\mathbf{b}} = \begin{bmatrix}293.7 & -934.7 & 10.9\end{bmatrix}^T$. For short range tossing on a running conveyor $\bar{\mathbf{b}} = \begin{bmatrix}281.4 & -982.1 & 11.7\end{bmatrix}^T$.

Table 4.5: Repeatability of **short range** tossing, expressed by the standard deviation and precision. The standard deviation of the rest-pose after tossing σ_{x_b}, σ_{y_b}, and σ_{θ_b}, are computed for a 95% confidence level, based on 30 tosses.

Toss Result	**Unit:**	**Idle conveyor**	**Running conveyor**
σ_{x_b}	[mm]	17.7 ± 6.3	17.3 ± 6.2
σ_{y_b}	[mm]	11.5 ± 4.1	7.2 ± 2.6
σ_{θ_b}	[°]	10.5 ± 3.7	11.8 ± 4.2
$\Delta\left({}^{U}\mathbf{o}_B\right)_{\mathbf{x}}$	[mm]	104.7	61.4
$\Delta\left({}^{U}\mathbf{o}_B\right)_{\mathbf{y}}$	[mm]	55.0	29.9
$\Delta\boldsymbol{\theta}$	[°]	44.8	49.3

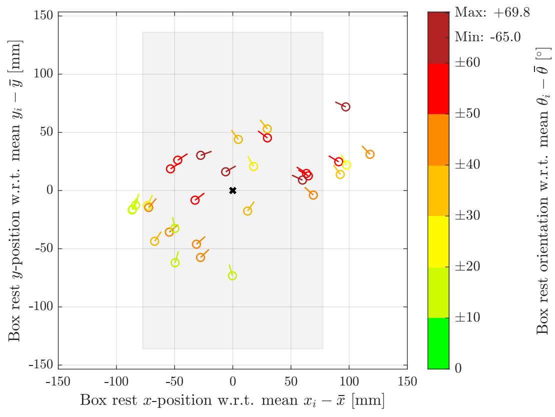


(a) Tossing on an **idle** conveyor.

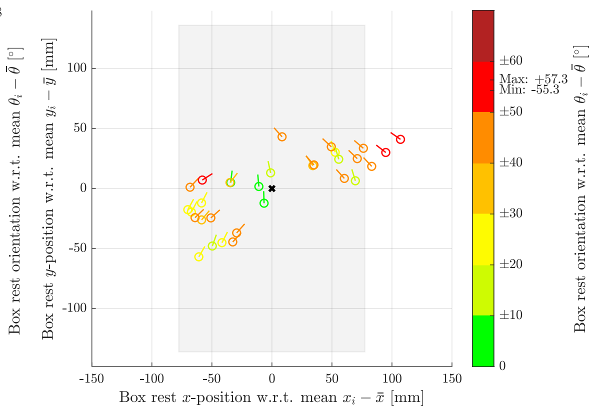


(b) Tossing on a **running** conveyor.

Figure 4.9: Box rest-poses $\mathbf{b}$ with respect to the mean $\bar{\mathbf{b}}$ after 30 **short range** tossing repetitions. The rest position of each toss is indicated by the circles, each rest position contains a line indicating the rest-pose orientation. For reference, the gray area displays Box 5.



(a) Tossing on an **idle** conveyor.



(b) Tossing on a **running** conveyor.

Figure 4.10: Box rest-poses $\mathbf{b}$ with respect to the mean $\bar{\mathbf{b}}$ after 30 **long range** tossing repetitions. The rest position of each toss is indicated by the circles, each rest position contains a line indicating the rest-pose orientation. For reference, the gray area displays Box 5.

Table 4.6: Repeatability of **long range** tossing, expressed by the standard deviation and precision. The standard deviation of the rest-pose after tossing σ_{x_b}, σ_{y_b}, and σ_{θ_b}, are computed for a 95% confidence level, based on 30 tosses.

Toss Result	**Unit:**	**Idle conveyor**	**Running conveyor**
σ_{x_b}	[mm]	62.8 ± 22.5	57.1 ± 20.4
σ_{y_b}	[mm]	34.9 ± 12.5	27.9 ± 10.0
σ_{θ_b}	[°]	44.8 ± 16.0	37.0 ± 13.2
$\Delta \left({}^U\mathbf{o}_B\right)_\mathbf{x}$	[mm]	204.5	104.7
$\Delta \left({}^U\mathbf{o}_B\right)_\mathbf{y}$	[mm]	145.1	55.0
$\boldsymbol{\Delta\theta}$	[°]	134.8	112.7

Long Range Tossing Repeatability Results

After 30 long range tossing repetitions on both an idle and running conveyor, the repeatability of the long range toss is determined using the OptiTrack measurement data. Figures 4.10a and 4.10b show the tossing results for long range tossing on an idle and running conveyor, respectively. The long range tossing repeatability results are found in Table 4.5. For long range tossing on an idle conveyor, the mean rest-pose was $\bar{\mathbf{b}} = \begin{bmatrix} 362.0 & -1446.8 & -10.4 \end{bmatrix}$. On a running conveyor, the mean box rest pose $\bar{\mathbf{b}} = \begin{bmatrix} 347.6 & -1483.5 & -14.6 \end{bmatrix}$.

Analysis and Interpretation of the Results

The goal of the experiment was to determine if a human demonstrated toss can be repeated, thereby leading to predictable outcomes. From the results of the short range tossing experiments, we conclude that Box 5 can be tossed with a positional precision of few centimeters, and acceptable orientations looking at the requirements for tossing in industry, presented in Section 4.1. Tossing on both the idle and stationary conveyor leads to box' rest-poses that distribute close to the mean tossing result. Hence, predictable robotic tossing can be achieved.

Looking at Tables 4.5 and 4.6 we see that tossing on a conveyor running at 0.4 m/s leads to better repeatability, compared to tossing on a stationary conveyor, as the box' rest poses distribute closer to the mean. This can be traced back to the fact that the box has a lower relative velocity, with respect to a running conveyor, while landing.

To put the results in perspective, we would like to make reference to the use case presented in Section 4.1). For tossing in the crossorter scenario, the box must end up within an area of 1300×2100 mm, while the box' rest pose orientation must be $\leq \|30°\|$ with respect to the mean, and the barcode must be visible. For tossing in the traysorter scenario, the box must consistently land in an area of 400×400 mm, under the only requirement that the barcode is pointing upwards. While the results up to now give a clear idea of the result of each toss, one might have trouble with translating these results to the final rest-pose of each box. To this end, the top view of the box' rest-pose after each short and long range toss are given by Figures 4.11 and 4.12, respectively. In

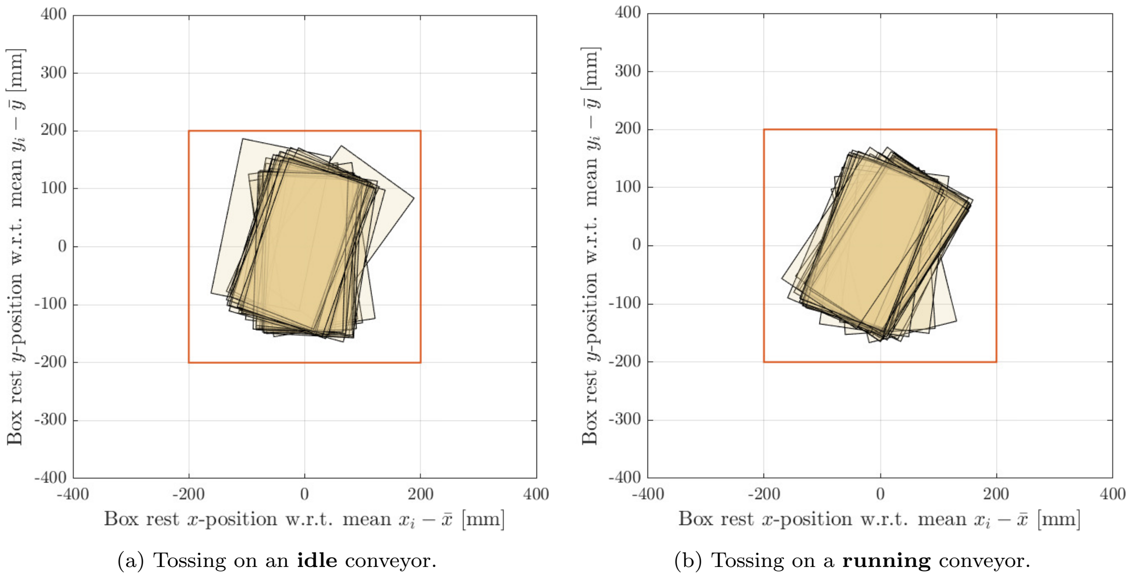


Figure 4.11: Box rest-poses $\mathbf{b}_i$ of 30 **short range** tossing repetitions, with respect to the mean rest-position, where the orange square depicts a TraySorter deck [73]. The running conveyor is moving in the vertical direction on the graph.

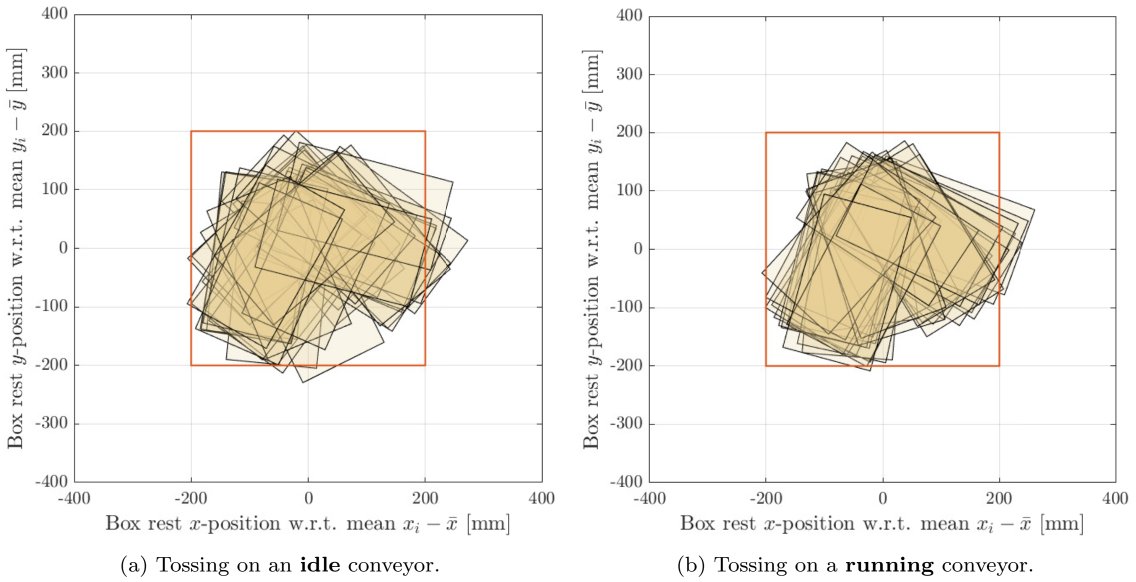


Figure 4.12: Box rest-poses $\mathbf{b}_i$ of 30 **long range** tossing repetitions, with respect to the mean rest-position, where the orange square depicts a TraySorter deck [73]. The running conveyor is moving in the vertical direction on the graph.

these figures, an orange square of 400 × 400 mm is drawn around the mean, which corresponds to the target area size of a tray in the traysorter process.

Based on the tossing requirements of the crossorter and traysorter scenarios, it seems that the short range toss is sufficiently precise for application in the traysorter- and crossorter infeed processes. However, future experiments must show how the box rest-poses deviate around the mean for a conveyor running at 1.5 m/s, which is the conveyor speed in the crossorter and traysorter processes. Although the box landed with its barcode up after each long range toss, the long range toss lead to less predictable rest position and orientation of the box. Both on an idle and running conveyor, the long range toss leads to box' rest-poses with too low precision for successful tossing in both the traysorter and crossorter processes. It seems that for longer range tossing, which requires higher tossing velocities, less predictable tossing can be achieved. Especially the box' orientation has a far larger deviation in its rest-pose. For future experiments, the requirements for proper tossing demonstrations can be tightened, to investigate if long range tossing can be achieved with more predictable tossing outcomes.

4.4 Pick Pose Sensitivity on Robotic Tossing

In the previous section, we have shown that predictable tossing can be achieved. We also know that the accuracy by which the result of a robot toss can be repeated, depends on multiple factors, such as the object's weight, dimensions, and shape, but also on the motion trajectory of a toss, and the pick pose of the suction gripper on the box. In industrial logistics processes, there are many conditions that change each time a parcel is tossed. For example, each pick-and-place operation, the robot might pick a different object, at a changed pick pose. Because of this, we want to investigate how controlled perturbations in the pick pose of the suction gripper on the box, affect the outcome of a toss. In this work, we refer to this as the pick pose sensitivity on robotic tossing. We define the sensitivity as the variation between the outcomes of successive tossing demonstrations, for changed pick poses, under the same measurement conditions. By tossing outcomes, as in Section 4.3, we refer to the rest position and orientation of the box after a toss.

The effect of a changed pick pose on a tossing result is interesting, especially since Vanderlande has examples of industrial settings where each object is the same, but automated picking technologies, have disturbances in the pick pose of the gripper onto an object. Consider an industrial setting where a UR10 robot unloads a container, filled with one type of box, and where each box has the same properties (mass, dimensions and inertia). In this setting, the pose of the suction gripper on the top surface of the box is not consistent. To explore how this affects the outcome of a toss, the reference trajectories and release timing will be kept constant.

In this section, first the goal is presented, see Section 4.4.1. Subsequently, the setup and procedure during the experiments is described in Section 4.4.2. Finally, the results and analysis of this study are discussed in Section 4.4.3.

4.4.1 Goal of the Sensitivity Study

By exploring the effect of inaccurate box picking, on tossing results, it can be explored which picking accuracy is required, and what tossing precision can be achieved for a given accuracy. The goal of the pick pose sensitivity exploration is to determine if reliable and predictable tossing can be achieved, while disturbances in the pick pose of the suction gripper on a box are introduced. Moreover, this exploration might help to get an idea of what are acceptable limits on deviations in the pick pose, and how each independent pick pose parameter affects the outcome of a toss. Furthermore, by analyzing the deviation and precision of tossing results for changed pick poses on one object, it can be explored whether tossing can be applied in the crossorter infeed or traysorter infeed processes, see Section 4.1. To this end, the following research question is answered:

Can a single reference tossing trajectory be used for automated tossing of an object that is picked on various positions and orientations, thereby leading to predictable tossing?

4.4.2 Procedure and Setup of the Sensitivity Study

During these experiments, the effect of a variable pick pose on the resulting pose of Box 5 after tossing, is explored. In the remainder of this work, these experiments are referred to as the sensitivity experiments. The procedure for these experiments is similar to the procedure that was followed through the repeatability experiments, presented in Section 4.3.2. Also, the same tossing trajectories are used during these experiments, this way, the results of the sensitivity experiments can be compared to the results of the repeatability, that were presented in Section 4.3.3. Due to time constraints, the sensitivity experiments are only carried out for tossing on an idle conveyor.

There are two differences in the procedure of the sensitivity experiments compared to the repeatability experiment: (1) the number of repetitions, and (2) that for each tossing repetition, a deviation in the pick pose is introduced. To introduce deviations in the pick pose, one setting is changed in the *Grab Box* mode (Mode 2, where the robot automatically picks the box, see Section 3.2.3). By this setting, we can change whether the robot picks the box at the center of its top surface, or if the box is picked with a predefined deviation in the pick pose. We define the pick pose in a similar fashion as we did for the box rest-pose, see (4.2). Since the box is picked by the suction gripper at the same height each time, we describe the pick position by only two parameters $\left({}^{B}\mathbf{o}_S\right)_{\mathrm{x}}$ and $\left({}^{B}\mathbf{o}_S\right)_{\mathrm{y}}$. Moreover, we assume the orientation of the suction cup frame S with respect to $[B]$ is only rotated over the $\mathbf{z}_B$ axis, hence,

$$ {}^{B}\mathbf{R}_S(\varphi) = \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.5} $$

where φ denotes the angle between $[B]$ and $[S]$. To this end, we express the pick pose of the i^{th} toss by

$$ \mathbf{p}_i = \begin{bmatrix} \left({}^{B}\mathbf{o}_S\right)_{\mathrm{x},i} & \left({}^{B}\mathbf{o}_S\right)_{\mathrm{y},i} & \varphi_i \end{bmatrix}^T. \tag{4.6} $$

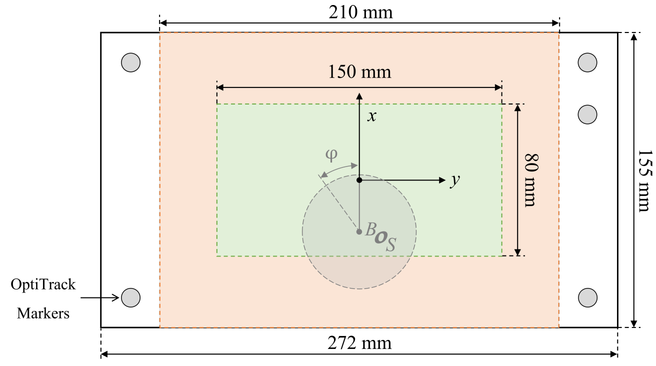


Figure 4.13: Schematic top view of Box 5, where the orange marked area is the area where the suction gripper is allowed to pick the box, and the green area depicts the area where the center of the suction gripper can pick the box.

We assume the pick pose deviation of parameters $\left({}^B\mathbf{o}_S\right)_\mathbf{x}$, $\left({}^B\mathbf{o}_S\right)_\mathbf{y}$, and $\boldsymbol{\varphi}$, in industrial applications, can be modelled as a Gaussian distribution. Therefore, we sample the pick poses from a multivariate Gaussian distribution with three variables. Gaussian distributions can be used to explain the statistical distribution of industrial processes in different disciplines of engineering and applied science, and even in describing many natural phenomena [77]. Consider the schematic view of Box 5, displayed by Figure 4.13. For the suction gripper to successfully pick the box, the center of the suction cup must be within the green marked area when the vacuum is activated. Therefore, the pick position must be within the range $\left({}^B\mathbf{o}_S\right)_\mathbf{x} = [-40, 40]$ mm, $\left({}^B\mathbf{o}_S\right)_\mathbf{x} = [-75, 75]$ mm. To study how deviations in the pick pose affect the tossing outcomes, we consider the full range of possible pick positions to pick Box 5. By doing so, we can explore the limits on the pick position parameters for Box 5. Moreover, the range for the pick orientation is chosen to be $\boldsymbol{\varphi} = [-30, 30]$ °, under the assumption that this will noticeably affect the tossing outcome. Let's assume the mean pick pose is given by $\bar{\mathbf{p}} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$. Based on the aforementioned ranges for each pick pose parameter, the covariance for $\left({}^B\mathbf{o}_S\right)_\mathbf{x}$, $\left({}^B\mathbf{o}_S\right)_\mathbf{y}$, and $\boldsymbol{\varphi}$, are chosen to be $3\sigma_{\mathrm{x_p}} = 75$ mm, $3\sigma_{\mathrm{y_p}} = 40$ mm, and $3\sigma_{\varphi_\mathrm{p}} = 30°$, respectively. As such, the pick poses are defined within the bounds for successful picking, with a probability of 99.7%.

Given the mean pick pose $\bar{\mathbf{p}}$, the covariance matrix

$$\Sigma = \begin{bmatrix} 3\sigma_{\mathrm{x_p}} & 0 & 0 \\ 0 & 3\sigma_{\mathrm{y_p}} & 0 \\ 0 & 0 & 3\sigma_{\varphi_\mathrm{p}} \end{bmatrix}, \tag{4.7}$$

and the number of samples $n = 125$, a Gaussian distribution of picking points is calculated using MATLAB [78] [79]. To start with, 125 Gaussian distributed pick poses are calculated. Figure

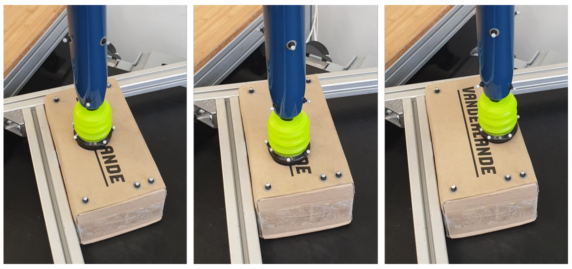
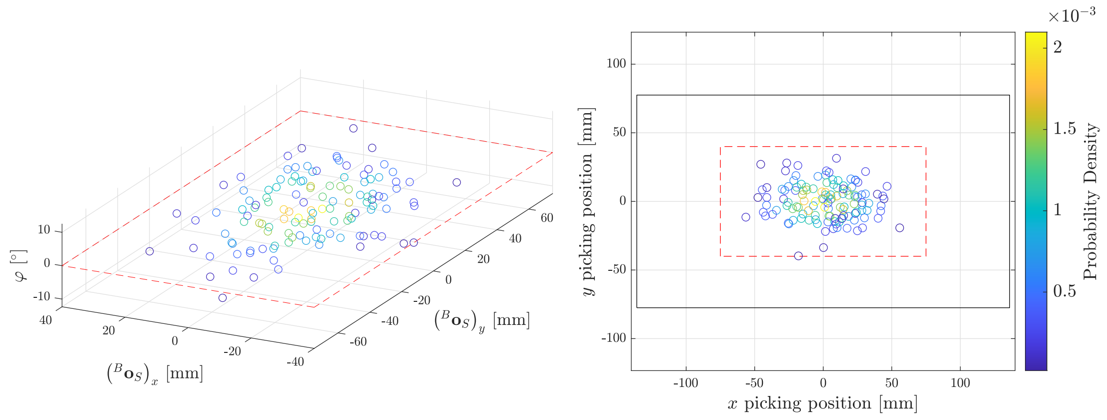


Figure 4.14: Three sample pick poses from the chosen Gaussian pick pose distribution on Box 5.



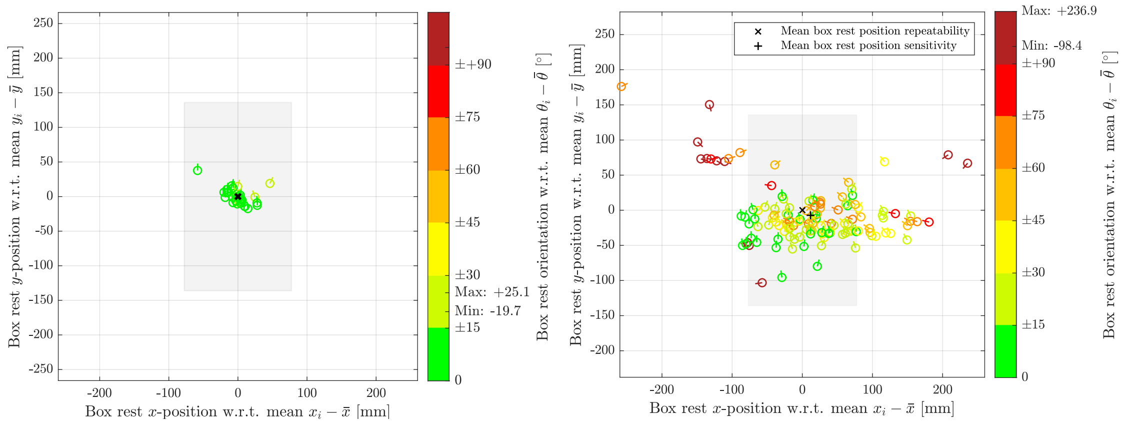
(a) 3D Gaussian distribution of pick poses. (b) Pick positions visualized on the box.

Figure 4.15: Visualization of the Gaussian distributed pick poses for Box 5, including probability density. The red dotted line corresponds to the pick position bounds, the black rectangle in (b) represents the border of Box 5.

4.15a depicts the Gaussian distribution of poses that are used as pick pose for the sensitivity study. Figure 4.15b depicts these points on a visualization of the box, with the corresponding probability density of the pick poses. Figure 4.14 shows an example of three different pick poses that were sampled from the Gaussian distribution.

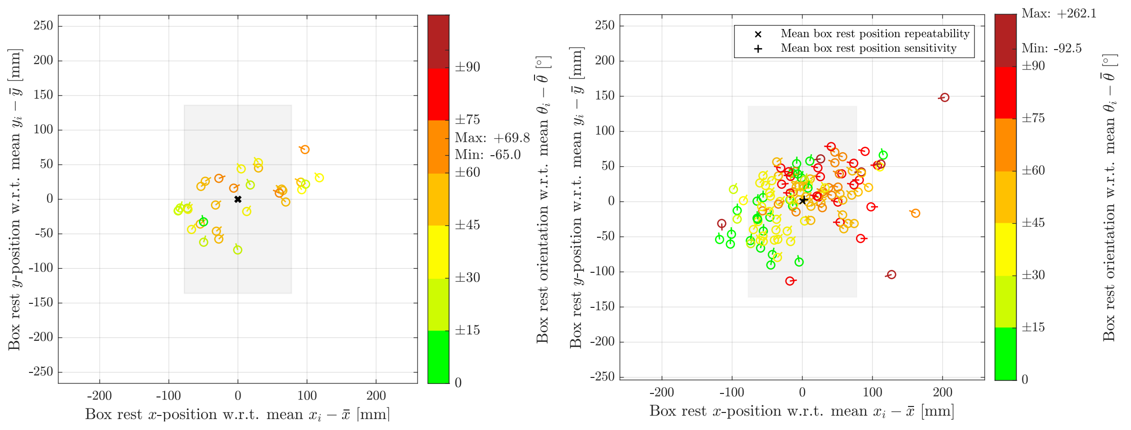
4.4.3 Sensitivity Study Results

The tossing outcomes of the sensitivity study, for the short and long range tossing trajectories, are depicted in Figures 4.16b and 4.17b, respectively. To show how the tossing outcomes are affected by controlled perturbations in the pick pose, the corresponding short and long range tossing outcomes of the repeatability study, are given by Figures 4.16a and 4.17a, respectively. Since the experiments are designed to analyze the predictability of robot tossing, the box' rest-pose orientations are not expressed by the circular mean, but by the complete angle a box rotated while landing. Note that the mean box rest-pose for repeatability and sensitivity, respectively indicated



(a) **b** for the repeatability experiments. (b) **b** for the sensitivity experiments.

Figure 4.16: Box rest-poses **b** with respect to the mean $\bar{\mathbf{b}}$ after 30 **short range repeatability** tosses on an **idle** conveyor (a), and 125 **short range sensitivity** tosses on an **idle** conveyor (b). The rest position of each toss is indicated by the circles, each rest position contains a line indicating the rest-pose orientation. For reference, the gray area displays Box 5.



(a) **b** for the repeatability experiments. (b) **b** for the sensitivity experiments.

Figure 4.17: Box rest-poses **b** with respect to the mean $\bar{\mathbf{b}}$ after 30 **long range repeatability** tosses on an **idle** conveyor (a), and 125 **long range sensitivity** tosses on an **idle** conveyor (b). The rest position of each toss is indicated by the circles, each rest position contains a line indicating the rest-pose orientation. For reference, the gray area displays Box 5.

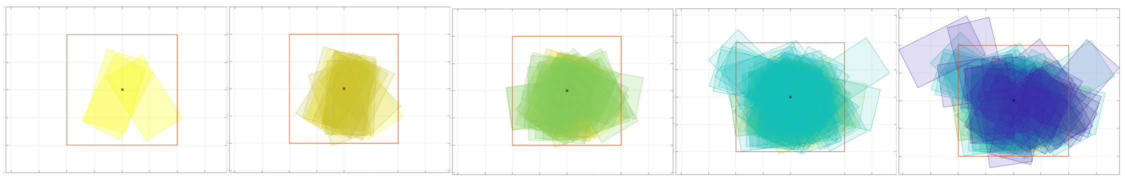
Table 4.7: Repeatability and sensitivity results for **short range** tossing, expressed in the standard deviation and precision. The standard deviation of the rest-pose after tossing $3\sigma_{\mathrm{x}_b}$, $3\sigma_{\mathrm{y}_b}$ and $3\sigma_{\theta_b}$, are computed for a 95% confidence level, based on 30 tossing samples.

		Short Range Toss		**Long Range Toss**	
Toss Result:	**Unit:**	**Repeatability**	**Sensitivity**	**Repeatability**	**Sensitivity**
σ_{x_b}	[mm]	17.7 ± 6.3	81.0 ± 14.2	62.8 ± 22.5	60.2 ± 10.6
σ_{y_b}	[mm]	11.5 ± 4.1	42.4 ± 7.4	34.9 ± 12.5	42.3 ± 7.4
σ_{θ_b}	[°]	10.5 ± 3.7	57.3 ± 10.0	44.8 ± 16.0	68.5 ± 12
$\Delta\left({}^{U}\mathbf{o}_B\right)_{\mathbf{x}}$	[mm]	104.7	493.5	204.5	321.1
$\Delta\left({}^{U}\mathbf{o}_B\right)_{\mathbf{y}}$	[mm]	55.0	279.4	145.1	261.4
$\Delta\boldsymbol{\theta}$	[°]	44.8	335.3	134.8	354.6

by the $\times$ and $+$ in the figures, are very close. For a quantitative comparison between the tossing outcomes, see Table 4.7. The standard deviation of the tossing results, for combined deviations in the pick poses, is obtained by 125 samples. To determine the sample standard deviation, we use the method as explained in Appendix I.

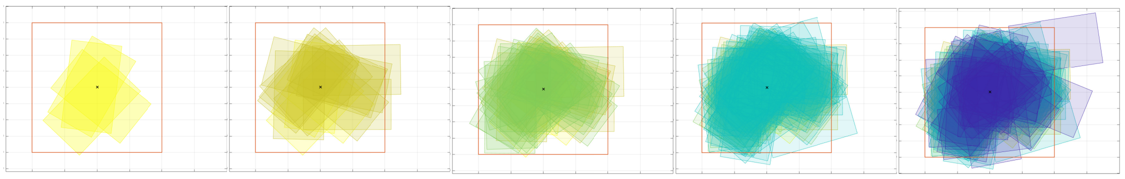
As expected, the controlled perturbations in the pick pose of a box, significantly affect the predictability of a toss. For the sensitivity results, the boxes end up in a larger area, and especially the deviation in the rest-pose orientation of the boxes increases significantly. For short range tossing with unchanging pick poses, the box' rest-positions had a deviation of no more than 10 cm, and the box would not rotate more than 25.1° with respect to the mean. From the sensitivity results, we observe that the box ends up to 30 cm from the mean rest-position, and that the box rotates up to 236.9° with respect to the mean.

Although the wide range of pick poses lead to unpredictable tossing, many tosses resulted in a box rest-pose that does lay close to the mean rest-pose of the repeatability experiments. This indicates that predictable tossing might be possible, even though the pick pose is not constant. This raises the question of how precise the robot must pick the box in order to perform predictable tossing operations. In Appendix J.1, an effort to determine for which pick pose precision the robot is able to perform predictable tossing, is presented. In this analysis, we divided the sampled pick pose distribution to explore what acceptable deviations in the pick pose are, for which robot



(a) $[\mathbf{0}, \frac{\mathbf{1}}{\mathbf{2}}\sigma]$ (b) $[\mathbf{0}, \frac{\mathbf{3}}{\mathbf{4}}\sigma]$ (c) $[\mathbf{0}, \sigma]$ (d) $[\mathbf{0}, \mathbf{2}\sigma]$ (e) $[\mathbf{0}, \mathbf{3}\sigma]$

Figure 4.18: Box' rest-poses for various ranges of deviations in the pick pose, for **short range** tossing. The orange square depicts the target area size of 400 × 400 mm in the traysorter process.



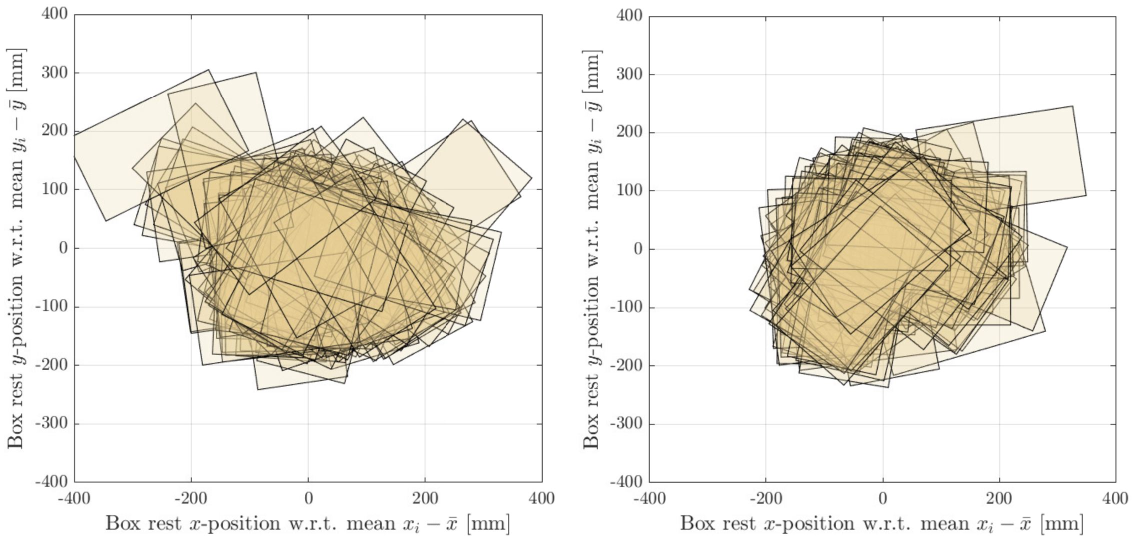
(a) $[\mathbf{0}, \frac{\mathbf{1}}{\mathbf{2}}\sigma]$ (b) $[\mathbf{0}, \frac{\mathbf{3}}{\mathbf{4}}\sigma]$ (c) $[\mathbf{0}, \sigma]$ (d) $[\mathbf{0}, \mathbf{2}\sigma]$ (e) $[\mathbf{0}, \mathbf{3}\sigma]$

Figure 4.19: Box' rest-poses for various ranges of deviations in the pick pose, for **long range** tossing. The orange square depicts the target area size of 400 × 400 mm in the traysorter process.

tossing leads to predictable outcomes. To this end, we presented the sensitivity results, that the box' rest-poses with large deviations in the pick pose can be distinguished from tossing outcomes with smaller deviations in the pick pose. The tossing outcomes for various ranges of deviations in the pick pose are depicted in Figures 4.18 and 4.19, for short and long range tossing, respectively.

Here, we use σ to denote each pick pose parameter covariance $\sigma_{\mathbf{x_p}}, \sigma_{\mathbf{y_p}}$, and $\sigma_{\varphi_\mathbf{p}}$. Hence, as explained in Appendix J.1, for the tossing outcomes that had a pick pose deviation in the interval $[0, \frac{1}{2}\sigma]$, we mean the ranges for each pick pose parameter are $\left({}^{B}\mathbf{o}_S\right)_{\mathbf{x}} = [-6.67, 6.67]$ mm, $\left({}^{B}\mathbf{o}_S\right)_{\mathbf{y}} = [-12.5, 12.5]$ mm, and $\boldsymbol{\varphi} = [-5°, 5°]$. It must be emphasized that there are only few pick pose samples with very small deviations ($< \frac{1}{2}\sigma$) in all pick pose parameters. However, in the results presented here, it is clearly visible that if deviations in the pick pose increase, tossing becomes less predictable. This conclusion applies to both short and long range tossing of Box 5. Consider Figure 4.18, for short range tosses, pick pose deviations up to $\frac{3}{4}\sigma$ in each pick pose parameter, lead to tossing results that are very close to comply with the traysorter requirements. Also, the deviation in rest-pose orientation is close to the allowable orientation precision for crossorter tossing. For long range tosses with deviations in the pick pose, the positional precision is somewhat comparable to the repeatability results, for deviations in the pick pose parameters up to σ. For the orientation, however, pick poses with deviations in the pick pose up to $\frac{3}{4}\sigma$ lead to much larger deviations in the box' rest-pose orientation. In Figure 4.19 the results from the sensitivity exploration for each range of deviations are shown with respect to the target area in the traysorter scenario. The long range tossing example is not precise enough to lead to reliable tossing results for both the crossorter and traysorter processes. To determine the allowable pick pose deviation for tossing, such that minimal deviations in the box' rest-pose are introduced, more experiments must be carried out.

In the analysis presented in Appendix J.1, where we analyze for which deviations in the pick pose predictable tossing can be achieved, we did not take into account how each pick pose parameter independently affects the outcome of a toss. If we explore this, we can make a better estimation of how precise the box must be picked, to achieve predictable tossing. To this end, in Appendix J.2, we present an analysis in an effort to explore how each individual pick pose parameter in the chosen sample distribution affects the box' final rest-pose.



(a) **Short range** tossing results. (b) **Long range** tossing results.

Figure 4.20: Box rest-poses $\mathbf{b}$ with respect to the mean $\bar{\mathbf{b}}$ after 125 tossing (sensitivity) repetitions on an idle conveyor.

The results presented in Appendix J.2, show that for the short range toss, large deviations in the pick orientation ($\varphi > 6.4°$), lead to unpredictable tossing outcomes. For deviations in the pick orientation that are $\varphi < 1.3°$, the tossing outcomes are significantly more precise. The pick orientation seems to significantly affect both the box' rest-position and orientation. Moreover, it seems that larger deviations of pick pose parameter $({}^{B}\mathbf{o}_S)_\mathbf{x}$, the box' rest-position in $({}^{U}\mathbf{o}_B)_\mathbf{y}$ decreases. Similarly, it seems that the effect of deviations in pick pose parameter $({}^{B}\mathbf{o}_S)_\mathbf{y}$ mainly results in lower precision of tossing outcomes in the box' rest-position in $({}^{U}\mathbf{o}_B)_\mathbf{x}$. For the long range toss, these effects are far less obvious. Therefore, unfortunately, the effects are not clear enough to base conclusions on. However, the obtained results help to understand if there is a clear mapping from error in the output space and allowable interval in the input space. Based on the analysis, we estimate that the pick pose parameters should be in the ranges of $\left({}^{B}\mathbf{o}_S\right)_\mathbf{x} = [-10, 10]$ mm, $\left({}^{B}\mathbf{o}_S\right)_\mathbf{x} = [-15, 15]$ mm, and $\boldsymbol{\varphi} = [-2.5, 2.5]$ °, to perform predictable tossing. To confirm this, however, a new set of sensitivity experiments is required, for pick poses sampled from a Gaussian distribution with pick pose parameter covariance matrix, see (4.7), with $3\sigma_{\mathrm{x_p}} = 10$ mm, $3\sigma_{\mathrm{y_p}} = 15$ mm, and $3\sigma_{\varphi_\mathbf{p}} = 2.5°$, respectively.

In Figure 4.20, all box' rest-poses after tossing are depicted. Looking at the use case applications for robotic tossing, as discussed in Section 4.1, boxes can be tossed to target areas precisely. However, it is challenging to toss Box 5 such that the rest-orientation of the box is consistently within 30°. While for unchanging pick poses the tosses were close to precise enough to comply with the requirements for tossing in the traysorter scenario, for changed pick poses the resulting rest-poses are far less precise. For both the short and long range tosses, positional precision still remains acceptable (for example, for a crossorter scenario). Moreover, the tossing accuracy also remains acceptable, since the mean box' rest-pose of the repeatability results is close to those of the sensitivity results. Based on the results, we think that robotic tossing can be applied in an industrial setting, even when the pick pose is not exactly the same each toss. In addition, the traysorter deck [73] has a back wall and edges around the surface on which the box lands. This might make it easier to apply robot tossing in this scenario, since the box is constrained in how much it can move after it landed on the traysorter deck.

Chapter 5

Conclusion and recommendations

The primary objective of the conducted work was to develop a teleoperation system which enables human operators to effectively perform tossing demonstrations. In this chapter, conclusions are drawn based on the results, and recommendations for further research on this topic are discussed.

5.1 Conclusion

This thesis is divided into two research objectives: *system development*, where the design and implementation of the teleoperation system for human-demonstrated tossing is described, and the *applicability of robotic tossing*, where the reliability and predictability of a human demonstrated robot toss is assessed. In the following two sections, a reflection regarding these topics is given.

5.1.1 System Development

The research objective related to the development of a teleoperation system particularly focused on tossing operations is:

Develop a teleoperation system that enables operators to control the motion of a robot manipulator and activation of the suction gripper through hand motion.

A teleoperation system for research around robotic tossing has been developed using a commercially available leader device. Based on a market review, the HTC Vive Pro kit was chosen for as a leader device in the teleoperation system due to its good price-quality ratio in terms of accuracy, and its good integratability and portability. After design of the system's hardware configuration, one of the most challenging topics was to establish the communication framework between the hardware components. To overcome this challenge, a way was found to directly retrieve the data from SteamVR in the controller's implementation. Using C++ objects, pose measurements and button data of the controller device are used to generate a pose acceleration reference input. Using forward kinematics, this reference input was converted into joint acceleration inputs for the QP controller, which in turn computes velocity level control inputs $\dot{q}$ for the robot manipulator. Through implementing the QP controller as an FSM controller, with separate modes for each task, such as teleoperation, recording manipulation tasks, repeating recorded tasks, automatic box picking, and moving to predefined poses, tasks could be implemented in convenient sequences, that are easily modified. The final controller implementation lays a good foundation to perform robotic tossing experiments, and for further research into robotic tossing. The low-costs and high portability of the leader device, provided with the plug-and-play nature of the software installation through GitLab[1], allow any UR10 robot controlled through mc_rtc to be extended with the teleoperation system very quickly. This way, the teleoperation system can be exploited and improved by developers in other research laboratories, and it can be used for various manipulation tasks.

[1]For the GitLab group containing software developed for the H2020 project I.AM., see: `https://gitlab.tue.nl/h2020-i-am-project`

Moreover, the empirical data that is/can be generated through the teleoperation system can be used to build more accurate tossing prediction models, and to confirm numerical simulations of tossing outcomes.

5.1.2 Applicability of Robotic Tossing

Formally, the research objective related to the reliability and predictability of robotic tossing operations, as stated in Section 1.3, is:

Quantify the variability in tossing outcomes resulting from autonomously repeating the demonstrated tossing motion, potentially introducing controlled perturbations, to explore the predictability of robot tossing.

In Chapter 4, an exploration of the reliability and predictability of human demonstrated robot tossing is presented. The experiments were carried out with Box 5, which represents the average box in the total range of boxes handled by the use case processes presented in Section 4.1. Therefore, we assume that the results presented in Chapter 4, give a fair indication whether robotic tossing can be applied in logistic warehouses. We must emphasize, however, that the presented results are not representative for the complete range of boxes, as well as other types of objects (e.g. bags, envelopes). Moreover, Box 5 has rather rounded edges and surfaces, due to the many times the object has been tossed before. When compared to tossing a box with sharp edges and straight surfaces, the hypothesis is that this causes more variance in the outcomes of repeated tosses. We chose to use Box 5 for the experiments, assuming that the results will represent the worst case scenario of predictability in tossing outcomes. Therefore, we believe that the conclusions given below will be representative for a wider range of boxes, but to prove this, and to confirm our hypothesis, many more experiments must be carried out.

To determine if robotic tossing can be successfully applied in logistics, the outcome of a robot toss must be predictable. Through experiments, we studied if a repeated tossing demonstration performed by the UR10, leads to comparable tossing outcomes. One short range and a long range toss were repeated by the robot hardware on an idle and running conveyor. The results of these tossing experiments, presented in Section 4.3.3, show that repetitions of tossing trajectories result in very similar outcomes. For longer range tossing, the precision decreases. Using this teleoperation system, various tossing methods can be studied to determine when robotic tossing leads to predictable results, and when this is not the case. We also have seen that tossing on a conveyor running at 0.4 m/s leads to more repeatable tossing outcomes than tossing on an idle conveyor. This difference is expectedly caused due to the difference in relative velocity between the box and conveyor for both situations. All in all, the results show that performing predictable robot tossing using the teleoperation setup is possible, and that recorded tossing trajectories can be repeated, thereby leading to predictable tossing outcomes. The variance in the tossing outcomes, however, is strongly dependent on the tossing trajectory. We find that for tossing with relatively low velocities, where the angle between the bottom surface of the box and the conveyor is relatively small, the outcome of a toss can be repeated quite reliably. When tossing at higher velocities, where the box makes impact with a large angle between its bottom surface and the conveyor, the outcome of a toss is less predictable.

Vanderlande identified two processes for which robotic tossing is potentially interesting, for usage in industrial logistics processes. In Section 4.1 these processes were presented: the traysorter infeed and crossorter infeed. For the traysorter infeed, boxes need to be tossed with a precision of 400×400 mm, with any orientation and its barcode pointing up. The crossorter infeed requires a low positional accuracy of the box, its barcode may not point downwards, and its orientation may deviate within $60° \pm 5°$. Given the requirements, and looking at the short range tossing results, we showed that robotic tossing can be done with sufficient accuracy in the use case scenarios. For long range tossing, the positional precision is just below the requirement for the traysorter infeed, and the rotational precision is far below the requirements for the crossorter infeed. We must emphasize, however, that the experiments were not carried out for tossing on a conveyor running that is running on the same speed as in the use case processes. To prove if robotic tossing can be reliably used in this case, the experiments must be repeated for tossing on a conveyor that runs at a velocity of 1.5 m/s. Moreover, during the experiments we didn't perform tossing on real traysorter decks, see [73]. Tossing on these decks will result in different outcomes because of the different material and construction of the traysorter deck, compared to the conveyor. On top of this, the traysorter deck has edges around the target area, and a back wall, which in practice, we can toss against. This expectedly allows for more deviation in tossing onto the traysorter decks.

In automated pick-and-place processes in industry, the pick pose of an object is often computed through vision technology. The pick pose of the suction gripper of an object in industrial processes is not consistently in the center of the box, like was the case in the repeatability study. We were curious to learn more about the predictability of a toss when uncertainties to the process are introduced. Through another set of experiments, we explored the sensitivity of the outcome of a toss, related to the pick pose of the suction gripper on the box. The results of these experiments are interesting to explore how deviations in the pick pose parameters affect tossing outcomes. Logically, inaccuracies in the pick pose introduce even larger variance in the outcome of a toss. However, Chapter 4 showed that repetitions of tosses can lead to very similar tossing outcomes, and that for small perturbations in the pick pose, predictable tossing can still be achieved. Therefore, we conclude that the results presented so far, offer great perspective in the possibility to reliably apply robotic tossing in logistic warehouses. to determine the exact precision by which boxes must be picked, more research is required.

5.2 Recommendations

During this work, several improvements for the current teleoperation system came to mind, these are presented in the following Section 5.2.1. Moreover, based on the experience gained during the experiments and the post-processing of the results, some new ideas came to mind to study the predictability of robot tossing. This is presented in Section 5.2.2. Since this research project is part of a vision that aims to develop automated solutions for tossing, some research directions are proposed in Section 5.2.3.

5.2.1 Teleoperation System Improvements

As expected, the robot responds very smoothly to velocity control inputs while operating the robot using the teleoperation device. However, open-loop QP control in velocity mode leads to drift, causing constraints and input positions to shift over time. Especially after a tossing demonstration, large drift occurs between the estimated joint states and the actual robot states. Commanding the robot using joint position control eliminates drift. However, for an appropriate QP controller stiffness gain, such that the teleoperation system can effectively be used to demonstrate tossing, joint position control introduces vibrations in the robot motion, which makes it difficult to perform teleoperated tossing demonstrations. To address the issue of drift in velocity control mode, the mc_rtc logs of multiple teleoperation demonstrations are analyzed, these contain the joint positions and velocities of the robot at each time instance, as well as the commanded velocities. This reveals that the UR10 is not tracking the velocity command sufficiently, which might be the reason of the observed drift. An obvious solution to eliminate the drift is to enable closed-loop QP control. Unfortunately, a stable implementation of a closed-loop QP controller failed, since for high acceleration inputs, the robot shows unstable behavior. This behavior can be explained by the robot's control frequency of 125 Hz, which might be too low for these input motions. For this control frequency, due to the controller gain, the controller might try to over-correct itself when high velocity control inputs are given, which would indeed lead to unstable behavior of the robot manipulator. As a final solution to cope with the issue of drift, the control state is updated from the real state after every toss, thereby eliminating the drift introduced by that same toss.

For this work, the latter solution is regarded sufficient. However, for future improvements of the teleoperation system closing the control loop might lead to more repeatable tossing, as the motion trajectories are followed more closely, and this minimizes the difference between two tossing repetitions.

To further improve the teleoperation setup, different mappings between the leader device and robot could be explored. Currently, the reference pose that is computed from the control signals and measurements from the leader device is used as input for the reference position of the bottom of the gripper's suction cup. By changing the location of the frame that is controlled using the reference input pose, the system might be even more effective and intuitive to perform tossing demonstrations. To compare the various motion mappings, the experiments as described in Appendix H can be used.

5.2.2 Tossing Predictability

In this work, only indications about the reliability and predictability of robotic tossing are given. This is because experiments were only carried out using one box, for two different tossing trajectories, and only changing two conditions: the conveyor velocity and pick pose of the box. To prove if robot tossing can really be applied in the logistics processes that were presented in Section 4.1, more experiments must be carried out. To this end, I would propose to adapt the robot setup at the lab such that the crossorter infeed and traysorter infeed scenarios are imitated. For this, the conveyor speed must be increased to 1.5 m/s, and traysorter decks to experiment tossing on must become available. Moreover, tossing must be carried out using the variety of objects that represent the range of objects that are handled in both the crossorter- and traysorter infeed processes. Besides proving the possibility of robotic tossing, this way, we can also determine the potential of robotic tossing in industry, as the required time for a pick-and-toss operation can be measured and compared to the time required by the SIR system to perform a pick-and-place operation. Moreover, the allowable pick pose deviation for a wide range can be determined.

In order to perform predictable tossing, it is important to identify how predictable tossing can be achieved. By carrying out experiments using the developed setup, much more empirical data of robotic tossing can be generated using the developed system. By analyzing the results of successful robot tosses, one might find a relationship between successful robot tosses, and identify how predictable tosses must be generated. In this work, we defined the requirements for a successful tossing demonstration on trial-and-error by simple observation of what would lead to a good toss, and what not. By analyzing more data about how the box should be released, and how it should land, more accurate requirements could be defined to determine whether a tossing demonstration is successful or not.

5.2.3 Development Towards Automated Tossing

This project is part of a larger vision that puts effort in the transition from automatic pick-and-place operations, to automated pick-and-toss operations. Therefore, we would like to propose directions that can help with achieving this goal, in which the developed teleoperation system can prove to be beneficial. Using the developed setup, the robot can repeat tossing from human demonstration. However, since the range of objects in logistic warehouses, and the properties of these objects are not constant (varying mass, inertia, etc.), a more flexible and reliable approach is preferred.

Once the teleoperation system is improved, see Section 5.2.1, and we have established good methods to achieve predictable tosses, see Section 5.2.2, one way to toss a wide variety of objects to desired locations might be to define the complete range in subgroups. Based on the results presented in this work, there is hope that a large variety of objects can be tossed, leading to predictable tossing results. The problem is that for a large variety of objects, a large variety of tossing demonstrations is required, likely with different tossing strategies depending on the specific object type.

Coupling every single object to an individual tossing demonstration is inefficient. However, if tossing demonstrations can be defined for ranges of objects, it might be feasible to toss a wide range of objects, by coupling a predefined tossing motion to groups of objects which have similar properties.

However, we doubt the reliability of such an implementation, in order to improve the adaptability of the robot, robot skill learning approaches can be used. This can be a powerful tool for enhancing and accelerating learning in robotics [80]. Robot skill learning encompasses a wide range of techniques and frameworks, the most popular are [81]: learning from demonstration and reinforcement learning. For teaching robots how to toss, the developed teleoperation systems can be advantageous for collection of high-quality demonstrations [82]. The TU/e impact-aware robotics database might prove useful to learn tossing from the empirical data, which makes the implementation cost less time, and gives the database more practical application value. The hypothesis is that, parametrization, learning algorithms, and development of dynamic motion primitives can contribute to the modeling, perception, and control of autonomous robotic tossing operations.

The process of learning from demonstrations (LfD) profits from situations where it might be easier for a teacher to demonstrate a desired behavior, rather than to engineer it manually. There are many publications available in which Learning from Demonstration (LfD) is explained, together with advantages and limitations of certain methods [81] [83] [84]. Some methods attempt to learn to replicate the desired behavior directly, a process which is called *behavioral cloning* [85]. Other methods rely on using demonstrations to learn the hidden objectives of the desired behavior. These methods are referred to as *inverse optimal control* [86], or *inverse reinforcement learning* [87]. An important distinction can be made between various methods are model-based approaches, and model-free approaches. In model-free approaches, the system dynamics are only implicitly encoded in the policy learned. Many learning approaches are successfully implemented by learning skills at trajectory level, instead of the control commands level. In model-free approaches, it is considerably more challenging to apply learning approaches, since it is often difficult to learn the system dynamics of a real robotic system, for instance for tasks involving impacts. We did not find literature to apply LfD particularly to teach robots to toss autonomously. In reinforcement learning, the robot learns by itself using guidance provided by a reward function [39]. However, this usually requires a very high number of demonstrations to learn a task: for instance, in [88], 3000 hours of robot interaction time was needed to learn hand-eye coordination for robotic grasping.

For teaching robots how to toss novel objects autonomously, we would suggest research into model-based learning from demonstration methods.

Bibliography

[1] N. Mikušová, Z. Čujan, and E. Tomková, "Robotization of logistics processes," *MATEC Web of Conferences, LOGI International Scientific Conference*, pp. 1–8, 2017.

[2] M. Klumpp, S. Bioly, and C. Witte, "Future Problems in Logistics Due to Demographic Change," *Hamburg International Conference of Logistics (HICL)*, pp. 51–68, 2014.

[3] J. Angeles, "Design Challenges in the Development of Fast Pick-and-place Robots," in *Romansy 19 - Robot Design, Dynamics and Control* (V. Padois, P. Bidaud, and O. Khatib, eds.), pp. 61–68, Springer Vienna, 2013.

[4] S. Kaewkorn, C. Joochim, P. Keeratiwintakorn, and A. Kunapinun, "Development of Pick and Place Delta Robot," *Advances in Intelligent Systems and Computing (AISC)*, pp. 475–486, 2020.

[5] R. Benotsmane, L. Dudás, and G. Kovács, "Trajectory optimization of industrial robot arms using a newly elaborated "whip-lashing" method," *Applied Sciences*, pp. 1–18, 2020.

[6] P. C. Huang and A. K. Mok, "A case study of cyber-physical system design: Autonomous pick-and-place robot," in *International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pp. 22–31, 2019.

[7] G. Tonietti, R. Schiavi, and A. Bicchi, "Design and control of a variable stiffness actuator for safe and fast physical human/robot interaction," *IEEE International Conference on Robotics and Automation*, pp. 526–531, 2005.

[8] M. Diehl, H. G. Bock, H. Diedam, and P. B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," *Lecture Notes in Control and Information Sciences*, pp. 65–93, 2006.

[9] H. Frank, N. Wellerdick-Wojtasik, B. Hagebeuker, G. Novak, and S. Mahlknecht, "Throwing objects - A bio-inspired approach for the transportation of parts," *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 91–96, 2006.

[10] Vanderlande, "Warehousing Innovative Systems: Smart Item Robotics," *https://www.vanderlande.com/systems/picking/smart-item-robotics/. Date accessed: 2021-09-02.*

[11] I.AM., "I.AM. Project," `https://i-am-project.eu/index.php`. *Date accessed: 2021-08-18.*

[12] A. T. Miller and P. K. Allen, "Graspit: A versatile simulator for robotic grasping," *IEEE Robotics and Automation Magazine*, pp. 110–122, 2004.

[13] J. Weisz and P. K. Allen, "Pose error robust grasping from contact wrench space metrics," in *IEEE International Conference on Robotics and Automation*, pp. 557–562, 2012.

[14] J. López, D. Pérez, I. Vaamonde, E. Paz, A. Vaamonde, and J. Cabaleiro, "Building a warehouse control system using RIDE," in *Robot 2015: Second Iberian Robotics Conference*, pp. 757–768, 2015.

[15] Z. Kootbally, C. Schlenoff, B. Antonishek, F. Proctor, T. Kramer, W. Harrison, A. Downs, and S. Gupta, "Enabling robot agility in manufacturing kitting applications," *Integrated Computer-Aided Engineering*, pp. 193–212, 2018.

[16] A. Zeng, S. Song, K. T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. C. Dafle, R. Holladay, I. Morena, P. Qu Nair, D. Green, I. Taylor, W. Liu, T. Funkhouser, and A. Rodriguez, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," in *IEEE International Conference on Robotics and Automation*, pp. 3750–3757, 2018.

[17] Y. Jiang, M. Lim, C. Zheng, and A. Saxena, "Learning to place new objects in a scene," *International Journal of Robotics Research*, pp. 1021–1043, 2012.

[18] M. Gualtieri, A. T. Pas, and R. Platt, "Pick and Place Without Geometric Object Models," in *IEEE International Conference on Robotics and Automation*, pp. 7433–7440, 2018.

[19] N. Marturi, M. Kopicki, A. Rastegarpanah, V. Rajasekaran, M. Adjigble, R. Stolkin, A. Leonardis, and Y. Bekiroglu, "Dynamic grasp and trajectory planning for moving objects," *Autonomous Robots*, pp. 1241–1256, 2019.

[20] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *IEEE Transactions on Robotics*, pp. 1049–1065, 2014.

[21] K. Benali, J. F. Brethe, F. Guerin, and M. Gorka, "Dual arm robot manipulator for grasping boxes of different dimensions in a logistics warehouse," in *IEEE International Conference on Industrial Technology*, pp. 147–152, 2018.

[22] C. Hernandez, M. Bharatheesha, W. Ko, H. Gaiser, J. Tan, K. van Deurzen, M. de Vries, B. Van Mil, J. van Egmond, R. Burger, M. Morariu, J. Ju, X. Gerrmann, R. Ensing, J. Van Frankenhuyzen, and M. Wisse, "Team delft's robot winner of the amazon picking challenge 2016," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 613–624, 2016.

[23] D. Morrison, A. W. Tow, M. McTaggart, R. Smith, N. Kelly-Boxall, S. Wade-McCue, J. Erskine, R. Grinover, A. Gurman, T. Hunn, D. Lee, A. Milan, T. Pham, G. Rallos, A. Razjigaev, T. Rowntree, K. Vijay, Z. Zhuang, C. Lehnert, I. Reid, P. Corke, and J. Leitner, "Cartman: The low-cost Cartesian Manipulator that won the Amazon Robotics Challenge," *Computing Research Repository (CoRR)*, pp. 1–8, 2017.

[24] D. Almeida, R. Ambrus, S. Caccamo, X. Chen, S. Cruciani, J. F. P. B. D. Carvalho, J. A. Haustein, A. Marzinotto, F. E. V. B., Y. Karayiannidis, P. Ögren, P. Jensfelt, and D. Kragic, "Team KTH's Picking Solution for the Amazon Picking Challenge 2016," *International Conference on Robotics and Automation (ICRA)*, pp. 53–62, 2017.

[25] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman, "Analysis and observations from the first Amazon picking challenge," *IEEE Transactions on Automation Science and Engineering*, pp. 172–188, 2018.

[26] RightHand Robotics, "RightPick™ System," *https://www.righthandrobotics.com/products/rightpick. Date accessed: 2021-08-03.*

[27] Inther Group, "GRIPP Robot Piece Picking," *https://www.inthergroup.nl/producten/orderpicken/gripp/. Date accessed: 2021-07-20.*

[28] A. Downs, Z. Kootbally, W. Harrison, P. Pilliptchak, B. Antonishek, M. Aksu, C. Schlenoff, and S. K. Gupta, “Assessing Industrial Robot agility through international competitions,” *Robotics and Computer-Integrated Manufacturing*, pp. 1–10, 2021.

[29] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, “TossingBot: Learning to Throw Arbitrary Objects with Residual Physics,” *IEEE Transactions on Robotics*, pp. 1307–1319, 2020.

[30] A. Nakashima, Y. Sugiyama, and Y. Hayakawa, “Paddle juggling of one ball by robot manipulator with visual servo,” *International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 1–6, 2006.

[31] Y. B. Jia, M. Gardner, and X. Mu, “Batting an in-flight object to the target:,” *International Journal of Robotics Research (IJRR)*, pp. 451–485, 2019.

[32] Y. Gai, Y. Kobayashi, and T. Emaru, “Motion control and optimization of a ball throwing robot with a flexible arm,” *12th International Conference on Motion and Vibration Control (MOVIC)*, pp. 937–945, 2007.

[33] J. S. Hu, M. C. Chien, Y. J. Chang, S. H. Su, and C. Y. Kai, “A ball-throwing robot with visual feedback,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2511–2512, 2010.

[34] W. August, S. Waldele, B. Hein, and H. Woern, “Accurate Object Throwing by an Industrial Robot Manipulator,” in *Australasian Conference on Robotics and Automation (ACRA)*, pp. 74–81, 2010.

[35] G. Niemeyer, C. Preusche, S. Stramigioli, and D. Lee, “Telerobotics,” *Springer Handbook of Robotics*, pp. 1085–1108, 2016.

[36] J. I. Lipton, A. J. Fay, and D. Rus, “Baxter’s Homunculus: Virtual Reality Spaces for Teleoperation in Manufacturing,” *IEEE Robotics and Automation Letters*, pp. 179–186, 2018.

[37] M. B. Schäfer, K. W. Stewart, N. Lösch, and P. P. Pott, “Telemanipulation of an Articulated Robotic Arm using a Commercial Virtual Reality Controller,” *Current Directions in Biomedical Engineering*, pp. 127–130, 2020.

[38] G. Gorjup, A. Dwivedi, N. Elangovan, and M. Liarokapis, “An Intuitive, Affordances Oriented Telemanipulation Framework for a Dual Robot Arm Hand System: On the Execution of Bimanual Tasks,” in *IEEE International Conference on Intelligent Robots and Systems*, pp. 3611–3616, 2019.

[39] K. Kamali, I. A. Bonev, and C. Desrosiers, “Real-time Motion Planning for Robotic Teleoperation Using Dynamic-goal Deep Reinforcement Learning,” *17th Conference on Computer and Robot Vision (CRV)*, pp. 182–189, 2020.

[40] J. Nakanishi, S. Itadera, T. Aoyama, and Y. Hasegawa, “Towards the development of an intuitive teleoperation system for human support robot using a VR device,” *Advanced Robotics*, pp. 1239–1253, 2020.

[41] S. Notheis, B. Hein, and H. Wörn, “Evaluation of a method for intuitive Telemanipulation based on view-dependent mapping and inhibition of movements,” *IEEE International Conference on Robotics and Automation*, pp. 5965–5970, 2014.

[42] D. Gong, J. Zhao, J. Yu, and G. Zuo, “Motion mapping of the heterogeneous master–slave system for intuitive telemanipulation,” *International Journal of Advanced Robotic Systems*, pp. 1–9, 2018.

[43] B. Siciliano and O. Khatib, "Telerobotics," in *Springer Handbook of Robotics*, pp. 741–754, Springer, Berlin, Heidelberg, 2008.

[44] M. A. Laribi, T. Riviere, M. Arsicault, and S. Zeghloul, "A new teleoperated robotic system for minimally invasive surgery: Modeling and identification," *International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 659–664, 2013.

[45] R. Eglash, "Broken metaphor: The master-slave analogy in technical literature," *Technology and Culture*, pp. 360–369, 2007.

[46] S. Traversaro and A. Saccon, "Multibody dynamics notation (version 2)," tech. rep., Eindhoven University of Technology, 2019.

[47] J. Nocedal and S. J. Wright, *16. Quadratic Programming.* 2006.

[48] J. Vaillant, K. Bouyarmane, and A. Kheddar, "Multi-Character Physical and Behavioral Interactions Controller," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1650–1662, 2017.

[49] U. Robots, "Ur10 - technical details," *https://www.universal-robots.com/products/ur10-robot/. Date accessed: 2021-07-05.*

[50] E. J. Colgate, W. Wannasuphoprasit, and M. Peshkin, "Cobots: Robots for collaboration with human operators," *International Mechanical Engineering Congress and Exhibition*, pp. 433–439, 1999.

[51] Fasth Fast-Berglund, F. Palmkvist, P. Nyqvist, S. Ekered, and M. Åkerman, "Evaluating Cobots for Final Assembly," *Procedia CIRP*, pp. 175–180, 2016.

[52] R. Bloss, "Collaborative robots are rapidly providing major improvements in productivity, safety, programing ease, portability and cost while addressing many new applications," *Industrial Robot: An International Journal*, pp. 463–468, 2016.

[53] Universal Robots, "RTDE C++ Interface," *https://sdurobotics.gitlab.io/ur_rtde/. Date accessed: 2021-10-03.*

[54] M. Borges, A. Symington, B. Coltin, T. Smith, and R. Ventura, "HTC Vive: Analysis and Accuracy Improvement," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2610–2615, 2018.

[55] P. Bauer, W. Lienhart, and S. Jost, "Accuracy investigation of the pose determination of a vr system," *Sensors*, pp. 1–17, 2021.

[56] VIVE European Union, "VIVE Pro Full Kit," *https://www.vive.com/us/product/vive-pro-full-kit/. Date accessed: 2021-09-02.*

[57] H. Wang, R. Mecham, and B. Zhang, "A Method Targeting Repair in Space: Tele-operating a Collaborative Robot with Virtual Reality," *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems, CYBER 2017*, pp. 1068–1071, 2018.

[58] N. Sanchez-Tamayo and J. P. Wachs, "Collaborative Robots in Surgical Research: A Low-Cost Adaptation," *ACM/IEEE International Conference on Human-Robot Interaction*, pp. 231–232, 2018.

[59] ValveSoftware, "OpenVR SDK," *https://github.com/ValveSoftware/openvr. Date accessed: 2021-11-02.*

[60] Joint Robotics Laboratory (JRL), "mc_rtc - overview," *https://jrl-umi3218.github.io/mc_rtc/index.html. Date accessed: 2021-01-08.*

[61] Joint Robotics Laboratory (JRL), "CNRS-AIST," *https://unit.aist.go.jp/jrl-2/index_en.html. Date accessed: 2021-09-27.*

[62] ROS, "Robot Operating System," *https://www.ros.org/. Date accessed: 2021-08-12.*

[63] ROS, "RViz Visual Tools," *http://wiki.ros.org/rviz/. Date accessed: 2021-08-12.*

[64] Joint Robotics Laboratory (JRL), "Github - mc_rtc," *https://github.com/jrl-umi3218/mc_rtc. Date accessed: 2021-10-05.*

[65] Valve Corporation, "SteamVR," *https://www.steamvr.com/en/. Date accessed: 2021-10-17.*

[66] Y. Abe, M. da Silva, and J. Popović, "Multiobjective Control with Frictional Contacts," p. 249–258, 2007.

[67] F. Bullo and R. M. Murray, "Proportional Derivative (PD) Control on the Euclidean Group," *European Control Conference*, pp. 1091–1097, 1995.

[68] K. Bouyarmane, K. Chappellet, J. Vaillant, and A. Kheddar, "Quadratic Programming for Multirobot and Task-Space Force Control," *IEEE Transactions on Robotics*, pp. 64–77, 2019.

[69] J. V. Steen, "QP control formulation and approaches CNRS and TU/e," tech. rep., Eindhoven University of Technology, 2020.

[70] K. Bouyarmane and A. Kheddar, "On weight-prioritized multitask control of humanoid robots," *IEEE Transactions on Automatic Control*, pp. 1542–1557, 2018.

[71] A. Escande, S. Miossec, M. Benallegue, and A. Kheddar, "A Strictly Convex Hull for Computing Proximity Distances With Continuous Gradients," *Robotics, IEEE Transactions on*, pp. 666–678, 2014.

[72] Vanderlande, "Crossorter," *https://www.vanderlande.com/systems/sorting/crossorter-2/. Date accessed: 2022-01-10.*

[73] Vanderlande, "Traysorter," *https://www.vanderlande.com/systems/sortation/traysorter/. Date accessed: 2022-01-10.*

[74] E. Mazareanu, "Statista, cross-border delivery: average package weight," *https://www.statista.com/statistics/974065/cross-border-delivery-package-weight-worldwide/. Date Accessed: 2022-01-10.*

[75] FormX, "Sculpture Block®," *https://www.formx.nl/clays/sculpture-block/index.php. Date accessed: 2021-07-27.*

[76] ISO, "ISO/IEC Guide 98-3:2008 - Uncertainty of measurement — Part 3: Guide to the expression of uncertainty in measurement (GUM:1995)," *https://www.iso.org/standard/50461.html. Date accessed: 2021-01-08.*

[77] I. Al-Nahhal, O. A. Dobre, E. Basar, C. Moloney, and S. Ikki, "A Fast, Accurate, and Separable Method for Fitting a Gaussian Function [Tips & Tricks]," *IEEE Signal Processing Magazine*, pp. 157–163, 2019.

[78] MathWorks Benelux, "MATLAB mvnrnd - Multivariate normal random numbers," *https://nl.mathworks.com/help/stats/mvnrnd.html. Date accessed: 2021-03-08.*

[79] Kotz, Samuel and Balakrishnan, Narayanaswamy and Johnson, Norman L., *Continuous Multivariate Distributions, Volume 1: Models and Applications.* 2019.

[80] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot Programming by Demonstration," *Springer Handbook of Robotics*, pp. 1371–1394, 2008.

[81] H. Ravichandar, A. Polydoros, S. Chernova, and A. Billard, "Recent Advances in Robot Learning from Demonstration," *Annual Review of Control Robotics and Autonomous Systems*, pp. 297–330, 2020.

[82] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation," in *IEEE International Conference on Robotics and Automation*, pp. 1–8, 2018.

[83] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, pp. 469–483, 2009.

[84] S. Chernova and A. L. Thomaz, "Robot learning from human teachers," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 28, pp. 1371–1394, 2014.

[85] M. Bain and C. Sammut, "A Framework for Behavioural Cloning," *Machine Intelligence*, pp. 1–37, 2000.

[86] A. Doerr, N. Ratliff, J. Bohg, M. Toussaint, and S. Schaal, "Direct loss minimization inverse optimal control," pp. 1–9, 2015.

[87] A. Y. and S. J. Russell, "Algorithms for inverse reinforcement learning," *17th International Conference on Machine Learning (ICML)*, p. 663–670, 2000.

[88] E. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *International Journal of Robotics Research*, pp. 421—436, 2018.

[89] OptiTrack, "Motion capture for robotics," *https://optitrack.com/applications/robotics/. Date accessed: 2021-07-05.*

[90] OptiTrack, "Prime 17w," *https://optitrack.com/cameras/prime-17w/. Date accessed: 2021-07-05.*

[91] OptiTrack, "Prime 22," *https://optitrack.com/cameras/primex-22/. Date accessed: 2021-07-05.*

[92] A. Naumann, J. Hurtienne, J. H. Israel, C. Mohs, M. C. Kindsmüller, H. A. Meyer, and S. Hußlein, "Intuitive Use of User Interfaces: Defining a Vague Concept," in *Engineering Psychology and Cognitive Ergonomics* (D. Harris, ed.), pp. 128–136, 2007.

[93] J. Hurtienne and L. Blessing, "Design for intuitive use - Testing image schema theory for user interface design," *International Conference on Engineering Design (ICED)*, pp. 1–12, 2007.

[94] J. Rasmussen, *The Human as a System Component.* Elsevier Science Inc., 1986.

[95] ISO, "ISO 9241-11: Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts," *https://www.iso.org/standard/63500.html. Date accessed: 2021-01-08.*

[96] F. Zijlstra and L. Doorn, "The Construction of a Scale to Measure Perceived Effort," tech. rep., Department of Philosophy and Social Sciences, Maastricht University, 1985.

[97] S. G. Hart and L. E. Staveland, "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research," in *Human Mental Workload* (P. A. Hancock and N. Meshkati, eds.), pp. 139–183, 1988.

[98] D. Ullrich and S. Diefenbach, "INTUI. Exploring the Facets of Intuitive Interaction.," in *Mensch & Computer: Interaktive Kulturen* (J. Ziegler and A. Schmidt, eds.), pp. 251–260, 2010.

[99] J. Hurtienne and A. Naumann, "QUESI - A questionnaire for measuring the subjective consequences of intuitive use," *Interdisciplinary College*, p. 536, 2010.

[100] Franka Emika, "Panda - Datasheet," `https://www.franka.de/robot-system/`. *Date accessed: 2021-07-05.*

[101] P. Nogueira, "Motion Capture Fundamentals A Critical and Comparative Analysis on Real-World Applications," pp. 1–12, 2012.

[102] D. C. Niehorster, L. Li, and M. Lappe, "The Accuracy and Precision of Position and Orientation Tracking in the HTC Vive Virtual Reality System for Scientific Research.," *i-Perception*, pp. 1–23, 2017.

[103] M. Suznjevic, M. Mandurov, and M. Matijasevic, "Performance and QoE assessment of HTC Vive and Oculus Rift for pick-and-place tasks in VR," *9th International Conference on Quality of Multimedia Experience (QoMEX)*, pp. 1–3, 2017.

[104] B. Fang, D. Guo, F. Sun, H. Liu, and Y. Wu, "A robotic hand-arm teleoperation system using human arm/hand with a novel data glove," *IEEE International Conference on Robotics and Biomimetics (IEEE-ROBIO)*, pp. 2483–2488, 2015.

[105] N. Miller, O. C. Jenkins, M. Kallmann, and M. J. Matarić, "Motion capture from inertial sensing for untethered humanoid teleoperation," *4th IEEE-RAS International Conference on Humanoid Robots*, pp. 547–565, 2004.

[106] F. Kobayashi, K. Kitabayashi, H. Nakamoto, and F. Kojima, "Hand/Arm robot teleoperation by inertial motion capture," *2nd International Conference on Robot, Vision and Signal Processing (RVSP)*, pp. 234–237, 2013.

[107] E. R. Bachmann, R. B. McGhee, X. Yun, and M. J. Zyda, "Inertial and magnetic posture tracking for inserting humans into networked virtual environments," *ACM symposium on Virtual reality software and technology*, pp. 1–9, 2001.

[108] O. Fukuda, T. Tsuji, M. Kaneko, and A. Otsuka, "A human-assisting manipulator teleoperated by EMG signals and arm motions," *IEEE Transactions on Robotics and Automation*, pp. 210–222, 2003.

[109] G. Du, P. Zhang, and D. Li, "Human-manipulator interface based on multisensory process via kalman filters," *IEEE Transactions on Industrial Electronics*, pp. 5411–5418, 2014.

[110] D. Roetenberg, P. J. Slycke, and P. H. Veltink, "Ambulatory position and orientation tracking fusing magnetic and inertial sensing," *IEEE Transactions on Biomedical Engineering*, pp. 883–890, 2007.

[111] S. Park, Y. Jung, and J. Bae, "A tele-operation interface with a motion capture system and a haptic glove," *13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pp. 544–549, 2016.

[112] B. Fang, F. Sun, H. Liu, and D. Guo, "A novel data glove using inertial and magnetic sensors for motion capture and robotic arm-hand teleoperation," *Industrial Robot: An International Journal*, pp. 155–165, 2017.

[113] J. Kofman, X. Wu, T. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *IEEE Transactions on Industrial Electronics*, pp. 1206–1219, 2005.

[114] G. Du, P. Zhang, J. Mai, and Z. Li, "Markerless Kinect-based hand tracking for robot teleoperation," *International Journal of Advanced Robotic Systems*, pp. 1–10, 2012.

[115] M. Z. Al-Faiz and A. F. Shanta, “Kinect-Based Humanoid Robotic Manipulator for Human Upper Limbs Movements Tracking,” *Intelligent Control and Automation*, pp. 29–37, 2015.

[116] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly, “Vision-based hand pose estimation: A review,” *Computer Vision and Image Understanding*, pp. 52–73, 2007.

[117] A. Jaju, A. Banerji, and P. K. Pal, “Development and evaluation of a telepresence interface for teleoperation of a robot manipulator,” *10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pp. 90–95, 2013.

[118] G. Song, S. Guo, and Q. Wang, “A tele-operation system based on haptic feedback,” *IEEE International Conference on Information Acquisition (ICIA)*, pp. 1127–1131, 2006.

[119] D. C. Niehorster, L. Li, and M. Lappe, “The Accuracy and Precision of Position and Orientation Tracking in the HTC Vive Virtual Reality System for Scientific Research.,” *i-Perception*, pp. 1–23, 2017.

[120] S. Fischer, M. Itoh, and T. Inagaki, “Screening Prototype Features in Terms of Intuitive Use: Design Considerations and Proof of Concept,” *Interacting with Computers*, pp. 256–270, 2015.

[121] S. G. Kwak and J. H. Kim, “Central limit theorem: the cornerstone of modern statistics.,” *Korean journal of anesthesiology*, pp. 144–156, 2017.

[122] D. R. Plane and K. R. Gordon, “A Simple Proof of the Nonapplicability of the Central Limit Theorem to Finite Populations,” *The American Statistician (TAS)*, pp. 175–176, 2012.

[123] J. L. Folks and R. S. Chhikara, “The Inverse Gaussian Distribution and its Statistical Application-A Review,” *Journal of the Royal Statistical Society: Series B (Methodological)*, pp. 263–275, 1978.

Appendix

A Vanderlande Innovation Lab

For implementation and validation of the proposed teleoperation system, the Vanderlande Innovation lab at the TU/e campus is available. Here, resources such as a robot setup, conveyor, and OptiTrack motion capture system are available. The setup in the lab, as depicted in Figure 1, can be used to perform experiments in an industrial-like setup. The robot manipulators can be programmed, via Ethernet connection, on a workstation (HP EliteDesk 800 G5 TWR, Core i7 vPro 9th Gen) running Linux with Ubuntu 20.04 as operating system.

A.1 Robot Manipulators at the Lab

The setup employs two robot manipulators, see Figure 1, which are both available for this project:

- the *Panda* by Franka Emika,
- and the *UR10* of Universal Robots.

Moreover, both robots equipped with the vacuum suction gripper of Smart Robotics. This gripper is equipped with a Venturi ejector located inside the gripper, which can generate a partially vacuum. The robots can therefore handle a wide range of objects, while enabling smooth and secure handling of goods. There are several differences between the two robots. First, the UR10 is a 6-DOF manipulator, whereas the Panda has one degree of freedom (DOF) more, therefore the Panda is kinematically redundant which makes it more flexible in dynamic environments. The UR10 has a higher payload than the Panda, 10 kg to 3 kg, respectively, and can thus manipulate heavier objects than the Panda. Furthermore, the UR10 has a larger reach. In terms of sensing capabilities, the difference is that the UR10 is only equipped with high resolution encoders, while the Panda has additional torque sensors.

A more detailed specification of the technical details of both the Panda and UR10 can be found in Appendix E.1 and E.2, respectively.

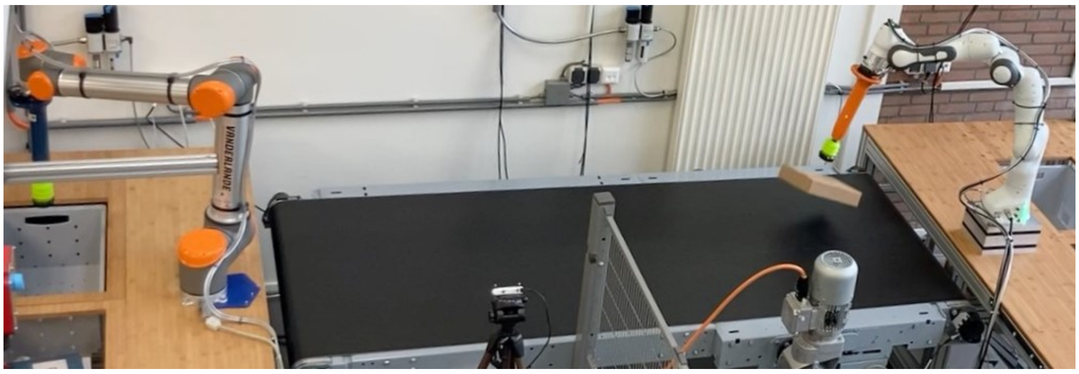


Figure 1: Vanderlande lab at the TU/e campus, with the UR10 (left) and Panda (right) robots.

(a) OptiTrack Prime 17W camera [90]. (b) OptiTrack Prime^x 22 camera [91].

Figure 2: OptiTrack cameras used in the setup at Vanderlande's innovation lab. Featuring 20 ultra high power LEDs, transmitting light at 850 nm IR, which is invisible to the naked eye.

A.2 OptiTrack Motion Capture System

In the Vanderlande Lab, there is an OptiTrack motion capture system in place, employing two Prime^{x}22 cameras and four Prime 17W cameras, see Figure 2, by which displacements of markers can be tracked at 360 Hz, and sub-millimeter accuracy [89]. The cameras are installed such that it can track the pose of the robot arm, lip of the suction cup, and boxes during manipulation, even while being tossed.

OptiTrack is based on recording specific 850 nm 70° IR light, which is either reflected by passive reflective markers or transmitted by active LEDs. The light reflected by the passive markers is transmitted by 20 high-intensity IR LEDs which the OptiTrack cameras are equipped with. By recording using multiple cameras, at high resolution (17 MP, 1664×1088 pixels) and frequency (360 Hz), at a wide angle, marker locations can be tracked at an accuracy up to 0.3 mm, depending on the calibration and scene [90]. Identified markers can be tracked by their location relative to each camera, which can be stored in a vector starting at the camera's origin and passing through the marker location. If at least two of these vectors are known, together with the relative position of each camera in the setup, a 3D position can be calculated. For this to allow accurate tracking, the cameras are calibrated each time the cameras are used. During calibration, the pose of each camera is defined in OptiTrack's inertial frame. While the cameras allow for grayscale footage, which is very helpful to identify unwanted reflections during calibration, markers are generally identified on-camera to avoid high bandwidth.

The OptiTrack system is controlled from its software platform *Motive*, where one can create rigid bodies from unique marker patterns. Such rigid bodies are represented by a frame defined with respect to the marker positions. Once a minimal amount of markers (at least 3) of identified markers fit a known pattern, the corresponding rigid body frame is fitted. Once a rigid body is created, these are automatically recognized and tracked. The measured rigid-body poses and marker positions are saved to a .tak file, which can then be exported to a .csv file. These .csv files will then be processed using MATLAB.

B Intuitive Use

One of the goals of this project is to develop the teleoperation setup such that it allows an operator to intuitively control a robot arm by hand motion, and thereby perform tossing operations. However, the term *intuitive use* (IU) is not well-defined in the sense of defining if, and to which extent, a system is intuitive. To develop an intuitive teleoperation system, a clear definition of intuitive use is required. In this section, IU is discussed to give a clear understanding about the meaning of the concept. Furthermore, we will investigate if there are methods to determine if a system is intuitive, and how an intuitive system is designed.

B.1 Definition

The concept of “intuitive” is commonly avoided by researchers because it is regarded as vague, ill defined or non-scientific. The interdisciplinary IUUI research group (Intuitive Use of User Interfaces) focuses on exploring the use of the term *intuitive use*, as well as to define it as a scientific concept [92]. Since IUUI tries to relate the IU approach to international usability standards, for this project, the choice is made to adopt the definition given by IUUI. This team of psychologists, computer scientists, engineers, and designers, defined the concept of *intuitive use* as follows: “A technical system is, in the context of a certain task, intuitively usable while the particular user is able to interact effectively, using subconscious application of prior knowledge.” [92]. This definition includes the concepts of *prior knowledge* and *subconscious application*, both of these concepts need to be clarified.

Prior knowledge refers to the knowledge, experience, and skills a user has at the start of learning a new topic or task. The prior knowledge a person has, may stem from different sources, divided over four levels [93]. The first level is innate knowledge, which is said to be universal to all humanity and is ‘acquired’ through the activation of genes, or during the prenatal stage of development. The second level is sensorimotor, which consists of general knowledge that is acquired very early in childhood, and from then on used continuously through interaction with the world. The third level refers to the knowledge specific to the culture an individual lives in, and can thus influence how people from different cultures approach technology. The fourth level is expertise, which is specialist knowledge, gained through experience. On each of the latter three levels (sensorimotor, culture and expertise) one might have specialist knowledge about using respective tools and technologies. At sensorimotor level these tools are primitive, at cultural level it refers to tools that are commonly used by humans (phones, lights, or cycles in the case of Dutch culture). At expertise level, there are tools that are used in one’s area of expertise, for example CAD tools for engineers. On each level of knowledge, IU (according to the above definition) can be assigned, as long as it is subconsciously applied by users. In the above definition, persons can only use intuitive processing, if they have had previous experience to exploit, it can therefore be said that things that a human can use intuitively, are those that employ features/characteristics that they have encountered before.

Subconscious application of prior knowledge, can be explained by the model of human data-processing of Rasmussen [94], which allows a designer to look at human-machine systems as one integrated system. This model is explained by the concept of a so-called *dynamic world model*, which refers to the “system” in humans that accounts for control of responses to the environment that are too fast to allow control by perceptual feedback; humans may have complex responses which can be released by very simple signals. To serve this purpose, it simulates the behaviour of the environment as well as the behaviour of the body; i.e., it simulates the interaction. When users respond to events or information that does not happen or appear in a familiar context, the dynamic world model also manifests itself. In the human data-processing model, Rasmussen distinguishes between a subconscious and a conscious processor. He states that perception is mapped subconsciously to the actual goal and states of the user, and also subconscious to the user’s *dynamic world model* stored in long term memory. This is stated to form the basis of efficient control of physical actions. However, when there is a mismatch between a user’s perception and dynamic world model, the conscious processor is activated. This conscious processor operates by controlling attention and is limited by the capacities of the user’s working memory and sequential processing.

Application of knowledge can become subconscious due to frequent exposure and reaction to stimuli in the environment. Knowledge at expertise level is not used as frequently as knowledge from culture or sensorimotor level. This suggests that knowledge from the lower levels is more likely to be applied subconsciously than expertise knowledge. As stated in the definition above, the subconscious application of knowledge is a precondition for IU. Therefore, it will be more likely for a user to interact intuitively, when lower level knowledge (innate, sensorimotor, and cultural) are involved.

B.2 Evaluating intuitive use

Setting *intuitive use* as a design requirement, involves the need for a method to measure if a design or system is intuitive. Measuring the intuitiveness of a system is complex since it regards the mental efficiency, and unconscious utilization of prior knowledge. Usability, of which IU is a sub-concept [93], is a (well-defined) measure of how well a specific user in a specific context can use a product/design. Analyzing how the indicators of usability relate to intuitive interaction, may provide insights in how IU can be evaluated. Usability involves measures for effectiveness, efficiency, and satisfaction [95]. In the ISO 9241-11, effectiveness is defined as the accuracy and completeness with which users achieve certain goals. Effectiveness can be indicated by error rates or the quality of goal achievement. Efficiency is defined as the relation between the effectiveness and the expended resources during the goal achievement. Expended resources can be indicated by measures like: task completion time, learning time, number of mouse clicks or key presses, material consumed, derived measures (such as financial cost), but also mental workload. Subconscious application of prior knowledge will affect efficiency since it reduces the workload from the conscious processor. Satisfaction refers to the comfort while and positive attitudes towards the use of the system, and can be assessed by a questionnaire.

Similar to the usability measures, IU has three indicators to measure intuitivity during human-robot interaction for a specific task: effectiveness, efficiency, and satisfaction. However, in usability measures, efficiency regards all resources (e.g. time, cognitive resources, material resources, energetic resources, financial resources, and number of steps to reach the goal) that the operator consumes. According to the definition of UI, only one resource of efficiency has direct correlation with IU: mental workload. Therefore, the three consequences by which IU can be assessed are: effectiveness, *cognitive* efficiency, and satisfaction. A precondition that has to be measured in order to assess IU is the probability of subconscious application of prior knowledge.

The definition of IU involves effective interaction, thus the indicators for effectiveness given by ISO 9241-11 will apply [95], such as the number of completed tasks and the proportion of errors. Moreover, effectiveness can be measured subjectively by using questionnaires to assess the percieved error rate and percieved achievement of goals. The cognitive efficiency can be measured either by objective means (e.g. physiological measures or response times) or by subjective means. For subjective measurements there are questionnaires available, like: Subjective Mental Effort Questionnaire (SMEQ) [96], and the NASA Task Load Index (TLX) [97]. The precondition can be assessed subjectively using questionnaires. For the probability of subconscious application of prior knowledge, so-called *technology familiarity* questionnaires can be used. Intuitive interaction measures also include satisfaction since users should be satisfied with using the system. Satisfaction for IU while interacting with a system can be evaluated by pragmatic and hedonic qualities in the INTUI questionnaire [98]. Moreover, Based on the theoretical framework presented by IUUI, a standardized questionnaire named “QUESI” was developed to measure the subjective satisfaction consequences of IU [99].

C Design Considerations

In this appendix, a discussion about the design considerations is given.

C.1 Control actions

One of the main challenges for this project is to design the telemanipulation system such that it is intuitive for the human operator. Controlling the system intuitively, while performing tossing operations, requires a system where the operator can quickly learn how to create control actions. Therefore, the system should be controlled in a way similar to how humans toss boxes. This raises the question *how can human tossing movements be transformed to “natural” control actions?* A box can be tossed in many ways, but for a human to toss a box with one hand, the human hand must move towards the box, and to grab onto it, finger movements (of the same hand) are required. Then, to move the box, the hand should hold the box. For tossing, the box must be accelerated and released, by finger movements. The natural way of tossing, will be used to create control actions, such that for an operator it is immediately clear how to move the manipulator, and how to operate the gripper. Therefore, the control of the robot manipulator motion will be done by the movements of only one arm. The gripper will be controlled by finger movements, which movements exactly is yet to be determined: opening your hand is similar to real human tossing, but adds significant complexity to the system, whereas only releasing a trigger with one finger is not exactly how a human releases a box. The robot manipulators both are equipped with a vacuum suction cup, extensibility is not considered because only this end-effector will be used for this project. However, the grasp of the object is therefore different, it is therefore interesting how tossing with a suction cup relates to tossing with hand grasps. Moreover, it might be interesting to develop a controller that is a replica of the suction gripper, which can be used to control the robot. This could increase the operator's feeling of telepresence.

C.2 Human variability

The setup must be operable for multiple humans, which have different characteristics. For intuitive control, the movements of the operator must be accurately followed by the robot manipulator. Therefore, in the design it must be considered how to deal with left- and right-handed persons or different limb lengths between operators. This raises the question: *how to deal with variability in human operators?* Does the setup have to be operable for left- and right-handed persons, or are there reasons to choose for a setup that can only be used with the right hand? And, how does the system deal with different body- and arm lengths, should the reference frame of the operator w.r.t the frame of the robot manipulator be calibrated for each person, and should this be manually provided to the system or automatically measured?

C.3 Anthropomorphism

By choosing an anthropomorphic robot arm, the system intuitiveness and operator's feeling of telepresence could be improved. However, there are two robot manipulators already available, from which the kinematic design cannot be changed. Therefore, other ways to improve the feeling of telepresence have to be studied: *how to enhance the feeling of telepresence?* The mapping of the operator movements to manipulator movements is important for the feeling of telepresence, the better the mapping, the better the feeling of telepresence. Taking this into account, the configuration of the robot manipulator and a human scaled-workspace of the telemanipulation setup are mandatory. One can study if the feeling of telepresence improves, if only the manipulator's end-effector must be controlled, or also the configuration of the robot. For example, when the human operator tosses underhand, does the robot also toss underhand, and similar for overhand tossing.

C.4 Operator position and viewpoint

Another important aspect is the viewpoint from the operator w.r.t to the robot manipulator. For intuitive control, we prefer a viewpoint close to the robot, that provides a complete overview of the surroundings of the robot. In the meantime, the operator must not be limited for moving in its own environment. Therefore, the following question must be answered: *what is a proper position and viewpoint of the operator w.r.t the robot?* Using VR, a virtual environment can be created, in which the operator feels like being *inside* the robot arm. However, the environment of the robot manipulator is uncertain since objects can enter within the kinematic range of the robot. Therefore, a VR environment is not desirable. To properly control the robot, while interacting with the environment, the human operator must directly see the robot. If the system is designed such, that the human operator can move to adapt her/his viewpoint, it can be practically analyzes which viewpoint is sensible. Moreover, the operator is more flexible to adapt to situations where the current viewpoint limits the operator's overview of the situation. Another pointer to decide which is an appropriate viewpoint, is which viewpoint provides a safe overview of the complete system. This is also dependent on the situation in the Vanderlande innovation lab at the TU/e campus.

C.5 Mapping input to robot reference

The mapping from control input to robot movements and control settings is an important aspect of the intuitive control. Finding a proper, intuitive mapping between human (hand-)motion and a non-anthropomorphic robot hand is challenging, due to the different joints, axes, and dissimilarities between the hand and vacuum suction cup. Therefore, the question *which mapping is most suitable to map human inputs to robot outputs* must be answered. To avoid difficulties in task completion of the human operator, a one-to-one (unscaled) position mapping is chosen to start with. For the mapping of input directions, which relate the movement directions of the operator to those of the robot manipulator, one can choose to map input directions in the operator (local) coordinate frame, or in the robot's coordinate frame. In case of a fixed mapping, in the robot's coordinate frame, together with the possibility of multiple viewpoints, may result in unexpected behaviour of the robot, or can at least be considered error-prone because the user is responsible to take the coordinate transformations into account. The choice of input mapping directions must be based on the development of a system in which the operator does not have to think about how to execute control actions. In order to develop a system by which a human operator can toss accurately, it can be useful to create a system which can filter human inaccuracies, such as holding a box at an angle. It also has to be determined where to define the base of the reference frame of the human operator. If the base is located at a fixed place in space, the operator is not able to change its viewpoint. If the operator has a moving base which is fixed on the operator, the operator can move to change the viewpoint. Another thing to deal with, is when the controller pose is forwarded to the robot. The system can be incorporated with a safety mechanism, such that the controller pose is only forwarded to the robot if an “enable” button is held. In case of such a button, one can also choose to program the telemanipulation system such, that the robot tool position follows the relative controller offset with respect to the initial state where tracking was enabled through the safety mechanism, while orientation tracking is absolute.

C.6 Interaction with environment

The robot manipulator will interact with its environment during human-operated manipulation, for example when gripping objects. This gives raise to the question: *how to deal with interaction between the robot and its environment?* If the operator controls the robot motion such that the desired trajectory of the robot goes through physical boundaries of the robot's environment, it must be made sure that the robot cannot damage its environment, and vice versa. A method to control the interaction forces is by impedance control, using this technique a desired damping and stiffness of the system can be chosen. However, more measures are required to make sure that the robot manipulator can only exert limited force on the environment. It also has to be determined what an appropriate value is for the maximum allowable force that can be applied to the environment. Similarly, the robot arm will also have to deal with different payloads, under which the robot should still follow the desired trajectory. During interaction of the robot with its environment, it would be valuable for the operator to be able to feel which forces the robot exert on the environment, and vice versa. For good tossing performance, having an indication of what objects weigh can help to estimate how to toss an object. However, the possibilities for force feedback depend on the available sensing in the robot manipulator, and require haptic feedback devices. For haptic force-feedback only a few haptic feedback systems are commercially available on the market. Because most of these devices are limited in their degrees of freedom (DOFs), have only a small workspace, or a low output capability, they do not enable an intuitive interaction with the remote environment. Moreover, most haptic interfaces that enable for force-feedback are fixed to the ground and therefore do not allow for any locomotion of the operator. A simpler method to offer haptic feedback functionality would be through controller vibration.

D System Requirements

In the proposed system, a UR10 robot is controlled by human motion, such that the operator can perform robot tossing demonstrations. It is important that the teleoperation system allows the operator to intuitively toss objects, to desired locations. This way, the teleoperation system can be used for effective trajectory generation of tossing trajectories for the robot. Besides this, the proposed teleoperation system is subject to a number of additional requirements, in this section these requirements are listed (R.1 - R.8)

R.1 The operator controls the robot manipulator through co-located teleoperation, in other words: the operator is present in the same room as the robot, and has a complete overview of the robot manipulator and its environment.

R.2 To enhance intuitivity of the system for tossing operations, the operator must be able to perform tossing operations using motion input that resembles human tossing motion. Therefore, the robot must be controlled in real-time by human hand motion. Additionally, human tossing motion may not be hindered due to the presence of hardware (e.g. cables) during operation.

R.3 The leader device must be a commercially available device, and must be able to track 6 DOF motion. The three translational motions: surge (forward/backward), heave (up/down), and sway (left/right). And the orientation described by three rotations: yaw (rotation over the normal axis), pitch (rotation over the transverse axis), and roll (rotation over the longitudinal axis).

R.4 Using the leader device, the operator must be able to control the vacuum suction gripper by two-step control, hereby either activating the vacuum gripper to pick objects, or deactivate the vacuum and activate the blow-off functionality to release objects. Hereby, the blow-off is normally off, and is only activated shortly when the vacuum is deactivated.

R.5 The teleoperation system must be equipped with a clutch (e.g. a button) to control when the follower device must imitate the leader device motion.

R.6 By controlling the robot manipulator through the teleoperation system, an operator must be able to toss Box 3, 4 and 5 (see Section 4.1) to locations outside the kinematic range of the robot.

R.7 For safety, the robot's controller must be implemented such that self-collisions and possible collisions between the robot and its environment are avoided.

R.8 The leader device/system must be able to track the operator's hand in a nominal workspace of minimal $2 \times 2 \times 2$ m, giving the operator some freedom to properly control the robot from the desired viewpoint.

E Available Robot Manipulators

E.1 Technical Specifications Franka Emika Panda

Franka Emika's Panda [100] is a cobot (collaborative robot), that is inspired on the mobility of the human arm, which results in an anthropomorphic design. The robot's payload is 3 kg and it's Cartesian velocity limit is 1.7 m/s, and features 7 DOFs, which makes the robot very agile. The kinematic reach is 855 mm, with a corresponding workspace coverage of 94.5%. Moreover, the robot arm has high resolution joint position encoders and link-side torque sensors incorporated in all 7 axes, allowing adjustable stiffness/compliance and advanced torque control. The robot's force and torque sensing resolution are < 0.05 N and < 0.02 Nm, respectively. The system has a pose repeatabillity of $< \pm 0.1$ mm (ISO 9283), and a path deviation[2] of $< \pm 1.25$ mm. The technical specifications of the Franka Emika Panda can be found in [100]. A summary of the specifications is given below.

[2]Based on ISO 9283 (Annex A): specified value refers to a workspace of $0.4 \times 0.4 \times 0.4$ m, centered at [0.515, 0.0, 0.226] m, with the Z-axis of the flange oriented parallel to earth-gravity and the elbow positioned upwards.

Table 1: Hardware- and performance specifications of Franka Emika's Panda robot [100].

Arm	
Degrees of freedom:	7
Payload:	3 kg
Max. reach:	855 mm
F/T sensing:	Link-side torque sensors in all 7 axes
Joint position limits:	• A1, A3, A5, A7: -166/166° • A2: -101/101° • A4: -176/-4° • A6: -1/215°
Weight:	∼17.8 kg
Moving mass:	∼12.8 kg
Control	
Supply voltage:	100 - 240 V_{AC}
Mains frequency:	47 - 63 Hz
Motion	
Joint velocity limits:	• A1, A3, A5, A7: -166/166 °/s • A2: -101/101 °/s • A4: -176/-4 °/s • A6: -1/215 °/s
Cartesian velocity limits:	Up to 2 m/s end effector speed
Pose repeatability:	$< \pm$ 0.1 mm (ISO 9283)
Path deviation:	$< \pm$ 1.25 mm
Force	
Force resolution:	< 0.05 N
Relative force accuracy:	0.8 N
Force repeatability:	< 0.15 N
Torque resolution:	< 0.02 Nm
Relative torque accuracy:	0.15 Nm
Torque repeatability:	< 0.05 Nm

E.2 Technical Specifications Universal Robots' UR10

Universal Robots' UR10 [49] is an industrial cobot. The payload of 10 kg and it's Cartesian velocity limit of 3 m/s allow for a wide range of applications in machine tending, palletizing, and packaging. The 6 DOF robot arm has a kinematic reach of 1300 mm, and no joint position limits, which allows for full workspace coverage. The UR10 is equipped with high resolution joint position encoders in all 6 joints, which allows for accurate position control. The system has a pose repeatability of ±0.1 mm (ISO 9283). The robot force sensing are estimations from the electrical currents in the joint actuators. The technical specifications of the Universal Robots UR10 can be found in [49]. An overview of the technical specifications is given by Table 2.

Table 2: Specifications of Universal Robots' UR10 [49].

Arm	
Degrees of freedom:	6
Payload:	10 kg
Max. reach:	1300 mm
F/T sensing:	Force in x-y-z directions and torque sensing in each joint
Joint position limits:	Base, shoulder, elbow and wrist 1, 2 and 3: ±360°
Weight:	∼33.5 kg
Control	
Supply voltage:	100 - 240 V_{AC}
Mains frequency:	47 - 440 Hz
Motion	
Joint velocity limits:	• Base and shoulder: ±120°/s
	• Elbow and wrist 1, 2 and 3: ±180°/s
Cartesian velocity limits:	1 m/s end effector speed
Pose repeatability:	< ± 0.05 mm
Force	
	Force Sensing Tool $x - y - z$, Joint Torque Sensing
Range	100.0 N, 10.0 Nm
Precision	5.0 N, 0.2 Nm
Accuracy	5.5 N, 0.5 Nm

E.3 Robot Manipulator Choice

The required hardware to develop a teleoperation system are: a leader device, a controller, and a follower device. This project is based at the Vanderlande Innovation Lab at the TU/e campus, here, two robot manipulators (follower devices) are readily available, of which one must be used for this project. In this section, both of these robot manipulators are presented. Furthermore, since there is a wide range of leader devices that can be used for the desired system, a review is presented about which (controller) devices can be used to control the robot manipulator are (commercially) available.

The two readily available robot manipulators for this project: are Franka Emika's *Panda*, and Universal Robots' *UR10*, see Figure 1. One of these robots will be implemented such that the end-effector will follow the leader device movements as closely as possible. Both robot manipulators are equipped with the vacuum gripper of Smart Robotics, which contains a Venturi ejector located inside the gripper, which generates a partially vacuum that allows the robots to handle a big variety of products. The robots can therefore handle a wide range of products, while enabling smooth and secure handling of goods. Moreover, both robots are capable of following a predefined trajectory by controlling joint position or velocity inputs and can be programmed on a Linux PC via Ethernet connection.

One of the two robot manipulators presented above, will be chosen for the proposed teleoperation setup. Here, a comparison is made which emphasizes the most important differences between the robots. An important difference between the robots is that the Panda is equipped with both high resolution encoders and torque sensors in each joint, whereas the UR10 only has high resolution encoders in its joints. The Panda is therefore able to provide haptic feedback to the operator when the robot interacts with objects or the environment, which enhances intuitivity while using the system. Moreover, the Panda is inspired on the mobility of the human arm, choosing this robot could lead to more intuitive tossing operations, than in case the UR10 is chosen. However, the Panda is a 7 DOF robot, which due to kinematic redundancy, complicates control compared to controlling a 6 DOF manipulator like the UR10. Additionally, the size of each robot is different, which results in a kinematic range of respectively 855 mm and 1300 mm, for the Panda and UR10. For tossing, the payload and tool velocity of the robots is very important. The UR10 is able to move objects of up to 10 kg at a speed of max. 3 m/s, whereas the Panda can only reach 1.7 m/s for objects up to 3 kg. Therefore, the UR10 is considered more suitable for tossing operations, and thus the UR10 is chosen to implement in the proposed teleoperation system.

F Review on Leader Devices

F.1 State-of-the-Art in Motion Capture

Market research into mocap systems reveals that there are two main categories mocap systems: optical and non-optical. The optical systems' category can be subdivided into: markerless, active (light emitting), or passive (reflective) systems. Non-optical mocap systems are subdivided into: inertial, mechanical, and magnetic mocap systems. Below, each of these system types are introduced and compared.

Optical Motion Capture

Active and passive systems use markers that are tracked by special cameras, markerless (or video) motion capture systems do not require markers or special equipment for tracking, but instead rely on software to track the subjects' movement [101]. Markerless motion capture systems use video and special software to track almost any subject. These systems provide good flexibility, but generally track subjects at a lower performance compared to marker-based solutions. Active mocap systems use special cameras that track the light that is emitted by LED markers. The emitted light provides that active systems are able to track markers at increased distance. Unlike passive- and markerless mocap systems, active mocap systems require the subject to carry wires or other electronic equipment. Passive mocap systems are the most accurate and commonly used in industry. These systems use reflective markers, that are tracked by infrared cameras. In these systems, each camera makes a two-dimensional image in which the position of the centroid of each marker is estimated. If multiple calibrated cameras capture a marker, a three-dimensional position can be obtained. Active and passive optical mocap systems can provide highly accurate 6D real-time tracking data of markers at high sample rates and in big workspace areas. However, sometimes these systems provide poor tracking, due to marker occlusion.

Inertial Motion Capture

Inertial mocap technology relies on inertial measurement units (IMUs), (bio)mechanical models, and sensor fusion algorithms. Using an IMU, linear acceleration and angular velocity along each of the three principal axes: pitch, roll and yaw can be measured. Sometimes the angular orientation can also be measured. The measurements are generated by integrating the acceleration measurements from the accelerometers and (a) gyroscope(s). Often, a magnetometer is used to measure the component of the earth's magnetic field in units of Gauss (G). The combination of the accelerometer and the magnetometer, is referred to as the digital compass of the IMU. The measurement outputs of the accelerometers, magnetometer and gyroscope can be converted in real-time to digital signals, which is often transmitted wirelessly to a computer. The computer can use this relative motion data to record the subject's full six DOF body motion. One of the main differences between optical and inertial motion capture is that optical systems can capture absolute positions, and inertial systems can only capture relative motion. To capture the subject's absolute position, external cameras and emitters/markers are required. Since IMUs can provide 6 DOF measurements to control the robot's end-effector position by relative motion inputs, this does not eliminate the possibility for the use of an inertial mocap system as leader device. An advantage of IMU's is that these devices can operate in very large areas and that marker occlusion, like in optical mocap systems, is not possible. A major disadvantage of IMUs is that they typically suffer from *dQuest*: an ever-increasing difference between the measured orientation and position output and the actual values. IMU's are widely available, devices range in price from low to high performing systems, with a bias instability (dQuest) of the gyroscope from $0.1°$/s to $0.001°$/h, and bias instability of the accelerometer from 100 mg to 10 μg.

Mechanical Motion Capture

Mechanical motion capture systems, often referred to as exoskeletons, directly track body joint angles by real-time measurements of the motion from the mechanical parts, articulated by the user's body motion [101]. In some cases, these devices can provide force-feedback or haptic input. Alternative to exoskeletons, robot arms can also be used as input device for the purpose of mechanical motion capture. Mechanical mocap systems have the advantage that these can provide force feedback to the user, but since the UR10 can not measure interaction forces, this ability can not be used. Disadvantages are that the user's motion is often constrained by the construction and extra weight, and the technology has no awareness of ground and direction in which the user is moving (and will thus not capture the complete tossing motion). Moreover, these systems are often costly. Hence, mechanical mocap systems will not be considered for controlling the motion of the teleoperator.

Magnetic Motion Capture

With magnetic mocap systems, it is possible to monitor 6DOF results of a motion capture session in real time. Magnetic motion capture systems can track a subject's position and orientation by placing receptors on the object's body, and by measuring the relative magnetic flux of three orthogonal coils on both the transmitter and each receiver [101]. The receptors are not occluded by nonmetallic objects, but metal objects in the environment (e.g. wiring, monitors or computers) cause magnetic and electrical interference. Some systems are even sensible to the building's structure (like steel reinforcing bars in concrete), making this a flaw in magnetic mocap systems. Moreover, magnetic motion capture systems do not provide a sufficiently large workspace. Due to these reasons, magnetic mocap systems are not suitable to serve as a leader device in the teleoperation system.

Acoustical Motion Capture

In acoustical mocap systems, three receptors are placed on site, and a set of sound transmitters are placed on the subject's main articulations. Then a characteristic set of frequencies is produced, by sequentially activating the emitters. The emitter's positions can then be calculated as the receptors pick up these signals, which calculate the emitters' positions in three-dimensional space [101]. Some problems with these systems occur, it is difficult to obtain a correct description of the data in real-time, the cables on the subject cause limitations of movement, and the amount of transmitters is limited, which limits the quality of the motion capture data. Moreover, acoustical systems are very susceptible to sound reflections or external noise. This makes it difficult to provide accurate real-time tracking data using this mocap technology and therefore, this technology is not suitable.

F.2 Market Review

Besides hand motion tracking, the operator must be able to use the leader device to somehow control the vacuum suction gripper for picking and releasing objects. There are two main methods to control the gripper: by two-step control (using a button, trigger or switch), or by measuring the operator's finger positions. The latter option controls the activation of the suction gripper by calculating if the hand is opened or closed. Measuring finger positions for gripper control can be done using commercially available gloves that capture high fidelity movements of all fingers, using OptiTrack markers, or by simple 1-axis measurements on one finger. A trigger, button, or switch can be operated by a commercially available controller device, or it can be mounted onto the operator's hand.

Based on the state-of-the-art in motion capture techniques, see Appendix F.1, we provide a market review in potential leader devices that can be used for the teleoperation system.

Handheld motion controllers

Commercially available motion controllers have become available since several game companies have developed high-accuracy and user-friendly motion capturing devices. Two of such controller devices are the HTC Vive[3] and Oculus Quest[4]. Review of the HTC Vive controller, showed that the device is suitable for robot control in case there is no requirement for accurate visual stimulation or self-motion through a visual world [102]. Since it has been showed that the HTC Vive performs better than the Oculus Quest [103], and both devices offer the same functionalities, the HTC Vive controller will be presented here.

The Vive's tracking technology uses two so-called Lighthouses, that contain laser emitters. These laser emitters send out horizontal and vertical infrared laser sweeps alternatingly, spanning 120° in both directions. The controller devices have photodiodes on their surface, which allows tracking 6D position of the controllers in space with a reasonable accuracy (less than 2 mm error) in a volume up to (4 m $\times$ 4 m $\times$ 4 m) [102]. In addition, the controllers provide extra inputs like a trackpad and trigger, and allow for feedback by vibration. Using the HTC Vive controllers, an intuitive teleoperation setup can be build, that enables the operator to smoothly control the end-effector of the robot manipulator with hand motions. Telemanipulation setups like these have already been successfully implemented [37] [38].

Motion sensors

Instead of using a handheld controller, the robot's motion and suction gripper can be controlled in alternative ways. Using traditional controller devices, the suction gripper is activated using buttons. However, a more natural way to control the release of an object could be to open your hand. Recently, with the rapid development of the motion sensors and vision-based techniques, it is believed that more natural and intuitive ways for robotic teleoperation are possible [104]. A common method to track human motion is using motion sensors such as: inertial motion tracking sensors [105] [106], electromagnetic motion tracking sensors [107], or electromyographic muscular activity sensors [108]. Inertial sensors are widely used because of the low cost, small size and energy efficiency. In [109], an effective method for the operator to focus on the manipulation task, rather than the required gesture for robot control is presented. Here, a position sensor and an inertial measurement unit (IMU) are used to track the 6D motion of the operator's hand. Using the sensory data, a robot could be controlled using the orientation and position of the hand. Moreover, electromagnetic sensors can also be used for motion tracking [110]. An inertial and magnetic measurement unit (IMMU) can be used for an accurate approach in estimating 6D positions, without external actuators or cameras. IMUs and IMMUs have also be incorporated in gloves, to control 6DOF robot manipulators [111] [112]. Gloves are often useful for robots employing dexterous hands, but can also be used to develop intuitive operation interfaces. Data gloves are often used in situations which require dextourous hand movements, and thus have to accurately capture the motion of the hand. In case of suction gripper control, the gloves only have to record if the hand is opened or closed, to activate (closed hand) or deactivate (open hand) the suction gripper. Commercially available motion capture data gloves are the CyberGlove[5], the Xsens gloves[6], and the Noitom HI5 VR gloves[7] compatible with the HTC Vive trackers[8].

Some of the presented leader devices may hinder natural human-limb motions due to cables. Since vision-based techniques only require physical markers, these techniques provide less hindrance to human motion. This should help the operator to concentrate on the task at hand, rather than on how to break the task into limited, gesture-like, commands that the human–robot system understands for each type of robot motion. Using vision systems with markers, a human-robot telemanipulation setup can be programmed such that human operators are able to communicate simultaneous 6DOF motion tasks to a robot manipulator [113]. This can also be done with the

[3]See: `https://www.vive.com/eu/product/#pro%20series`.
[4]See: `https://www.oculus.com/quest-2/`.
[5]See: `http://www.cyberglovesystems.com/`.
[6]See: `https://www.xsens.com/products/xsens-gloves-by-manus`
[7]See: `https://www.noitom.com/hi5-vr-glove`
[8]See: `https://www.vive.com/eu/accessory/tracker3/`

OptiTrack[9], which is already available in the Vanderlande Innovation lab at the TU/e campus. In [114], a robot manipulator teleoperation setup is presented by which Kinect[10] sensor data controls the robot. In this setup, an operator is able to control the 3D motion of the robot using hand positions, and the gripper configuration using hand gestures. The Kinect has also been used for the control of humanoid robots [115]. The Kinect tracking device avoids that there may be marker occlusion and identification. However, vision-based systems are often affected by varying light conditions, changing backgrounds, and a large clutter. Due to this, extracting relevant information from visual signals is difficult and computationally intensive. Furthermore, the occlusion of fingers to control the gripper signal may result in a non-observable problem that leads to a poor estimation of the hand pose [116].

Instead of mounting OptiTrack markers directly on the operator's fingers, the OptiTrack gloves by Manus[11] are a plug-and-play solution to track accurate finger data directly into the Optitrack control software *Motive*. In such system, the OptiTrack markers can capture the 6DOF hand motion. For gripper control, the gloves can track finger movements to capture if the operator's hand is open or closed.

Haptic devices

In case haptic feedback to the operator is required, haptic devices can be used. During control of a robot manipulator using a haptic device, the forces that act on the robot arm can be exerted to the stylus of the haptic device, so the operator can adjust the position and direction of the robot arm in the distance according to the haptic feedback. The use of feedback, however, is only possible if the robot manipulator is equipped with force sensors. A frequently used haptic device for the control of manipulators, is the OpenHaptics Touch X[12]. The device has 6 DOFs, that allows the operator to perceive forces related to the obstacle and environment in any direction [117] [118]. The device's workspace is 16 cm x 12 cm x 7 cm, and it can provide force feedback up to 3.3 N, its positional resolution is around 0.055 mm.

G Technical Details HTC Vive Pro Kit

The Vive Pro Kit, see Figure 3.3, comes with the following components: a head mounted display (HMD), link box, two VR controllers (2018 version), and two basestations 2.0. To ensure stable tracking and limit dQuest in the pose tracking, a third basestation was ordered separately.
Vive's tracking system operates through a so-called *outside-in* principle, where external tools are used to calculate the position of the system's equipment [119]. Vive estimates the positions of the HMD and VR controllers through a time difference [54]. The Vive system uses two (or three in this case) basestations (also referred to as lighthouses), which are stationary sensor units that continuously send out vertical and horizontal sweeps of infrared light in an alternating manner, at an angle of 120° per cycle in each direction, see Figure 3. Besides the sweeping plane of infrared light, in every cycle, a synchronization flash of infrared light is emitted, containing calibration parameters to correct the sweeping plane of light. The HMD and the VR controllers are equipped with photodiodes that indicate when the infrared light hits them. By detecting both these signals, given that the laser rotates at a constant angular velocity, the time difference between the signals can be used to estimate the angle α (see Figure 3) between the normal vector of the basestation and the photodiode. Using the estimated angle for both a horizontally and vertically rotating laser, and given the fact that each rigid body consists of multiple photodiodes, the absolute pose of each VR controller and the HMD can be calculated. A problem that is similar to calculating the pose of the HMD and VR controllers is the Perspective-n-Point (PnP) problem, for more details about the pose estimation principle, refer to [120]. In the calculations, multiple frames are involved [54] (see Section 2.2 for documentation about multibody notations), see Figure 4.

[9]See: `https://optitrack.com/`.
[10]See: `https://azure.microsoft.com/en-us/services/kinect-dk/`.
[11]See: `https://www.manus-vr.com/optitrack-gloves`
[12]See: `https://www.3dsystems.com/haptics-devices/touch`.

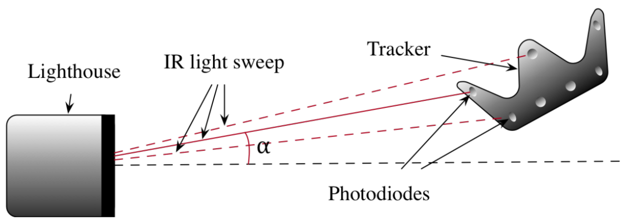


Figure 3: Illustration of the HTC Vive working principle, with α being the angle between a sweeping plane and the normal vector to the basestation (lighthouse) [54]. Note that in this figure, a Vive tracker is displayed instead of the VR controller that is used in the proposed teleoperation setup.

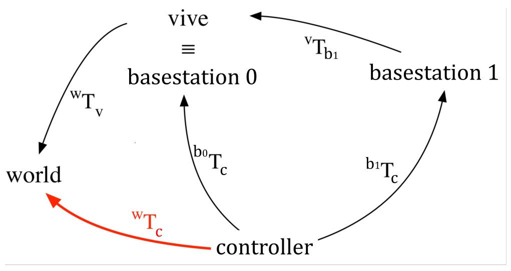


Figure 4: Schematic overview of the frames that are involved in the pose estimation of Vive's VR controller [54]. Here, ${}^{y}T_{z}$ denotes a rigid-body transform from frame y to z (which is equivalent to the pose of y in frame z) [46].

The frames of the basestations are indicated by b_i, with i the index of the basestation. The frame of the controller is indicated by c, and an auxiliary frame v, is selected during the calibration procedure to always be coincident with one of the basestation's frames. The frames allow the system to relate the poses of the basestations, since multiple basestations can be used. Moreover, the world frame w can be selected by the user, since the HMD is used to set the world frame during the calibration. The output that is used by the teleoperation system to generate the reference input for the controller is a rigid-body transform ${}^{w}T_{c}$ between a tracker frame and the world frame, represented by a red arrow in Figure 4. Detailed information about how the reference trajectory is generated, is presented in Section 3.2.2. The HMD and VR controllers are also equipped with inertial measurement units (IMUs), which are used to primarily update the HMD pose through deadreckoning (path integration), this allows for much higher update rates. Using the basestations, the error build-up that is inherent in deadreckoning is limited and corrected, based on inertial measurements. Based on this technology, the pose of the VR controller's can be estimated at a 120 Hz refresh rate, with sub-millimeter accuracy [54]. The average position's standard deviation for pose estimation of the static (not moving) device was found to be 0.242 mm. However, the dynamic tracking accuracy varies from few millimeters, but experimental results have shown that the tracking error can become up to a meter [54]. In [55], experimental research has shown that the position tracking accuracy of the system is few millimeters, when two lighthouses are used. It has also been proven, that the second generation of the HTC Pro system does not provide a horizontal reference plane, an inclination of 0.8° has been observed for two lighthouses. Major impacts on the overall accuracy are a poor calibration of the VR devices, and a changing hierarchy in the lighthouse system (when two or more lighthouses are used). These effects can lead to position errors of over one centimeter (up to 4 cm).

H QP Controller Parameter Choices

The QP controller has various parameters for which suitable values have to be found, such that the teleoperation system is suitable for demonstrating tossing operations. In this work, it has been assumed that human intuition and learning abilities can contribute to the rapid creation of motion trajectories for directed robotic tossing. The development of the teleoperation system enabled to rapidly demonstrate robotic tossing, thus generating motion trajectories for tossing. This experiment aims to improve the teleoperation system such, that the operator is able to generate motion trajectories quicker. Therefore, the goal of this experiment is to find parameter settings for which the operator is able to perform targeted robotic tossing demonstrations, with the least amount of required attempts.

H.1 Procedure and setup

In this experiment, the following parameters are tweaked:

- QP controller task stiffness gains k, see Section 3.3.2.
- Reference position scaling α, see Section 3.2.2.
- QP controller task weight ratio $w_{\text{pos}}/w_{\text{ori}}$, see Section 3.3.4.

Because this study is only meant to set reasonable values for the teleoperation system for demonstrating tossing, and the goal is not to find the ideal combination of parameter settings, it is assumed that the parameters do not mutually influence the tossing result. Therefore, the experiment is executed for each parameter seperately, in three sets (A, B, and C). The chosen parameter values that will be used during each round of the experiment are denoted in Table 3. During Set A, we vary the task stiffness, during Set B, we tweak the reference position scaling, and during Set C, the task weight ratio is tweaked. As can be seen in the table, the order of parameter values is randomized, to minimize the influence of potential learning effects on the outcome of the experiments. For each parameter, a default value is chosen that is used when that parameter is not varied (e.g. during experiment Set A, the values for scaling and task weight ratio are set at 2 and 100/6, respectively). The default values are chosen to be the mean of each range of parameter values. During experiment Set A, the task stiffness gain is varied in steps of 5 N/m, between a minimal value for the stiffness gain of $k = 25$ N/m, and a maximum value of $k = 45$ N/m. The maximum has been found to be the maximum stiffness gain for smooth control of the robot using the teleoperation setup. The maximum scaling for which the robot can be controlled smoothly was found to be $\alpha = 3$. For a scaling value $\alpha < 1$, teleoperated robot tossing was not effective. Therefore, during experiment Set B, the scaling is varied between $\alpha = 1$ and a $\alpha = 3$, in steps of

Table 3: The different parameter settings that will be checked during the experiments.

	Set A:	$\boldsymbol{k}$	**Set B:**	$\boldsymbol{\alpha}$	**Set C:**	$\boldsymbol{w_{\text{pos}}/w_{\text{ori}}}$
	A-1	40 N/m	B-1	3	C-1	100/10
	A-2	30 N/m	B-2	1	C-2	100/8
Defaults $\longrightarrow$	A-3	35 N/m	B-3	2	C-3	100/6
	A-4	25 N/m	B-4	1.5	C-4	100/4
	A-5	45 N/m	B-5	2.5	C-5	100/12

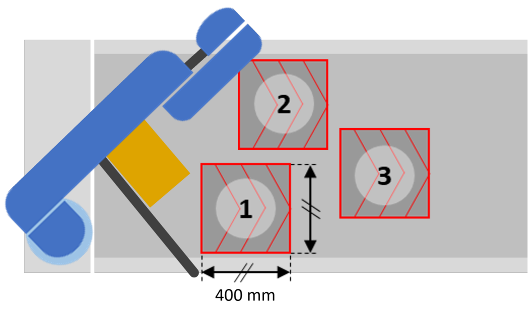


Figure 5: Schematic top view of robot setup with three tossing targets.

0.5. During experiment Set C, the task weight ratio is varied between a ratio of 100/4 and 100/12, with steps of 100/2. A ratio of 100/12 is the ratio that is found to be the maximum to keep the deviations on the position task small when the orientation error is large.

During the experiment, the operator is asked to toss boxes onto three predefined targets. These predefined targets are located on the conveyor in front of the robot, as displayed in Figure 5. Each target is a square area of 400 $\times$ 400 mm. The operator gets three different boxes: Box 3, Box 4, and Box 5, see Table 4.2. During each experiment set A to C, the parameter values are substituted in the controller, according to Table 3. For example, during experiment Set A-1, the value for $k = 40$ N/m, and the values for the scaling α and task weight ratio $w_{\text{pos}}/w_{\text{ori}}$ are set to default ($\alpha = 2, w_{\text{pos}}/w_{\text{ori}} = 100/6$ N/m). In another example, for experiment Set C-4, the value for the task weight ratio $w_{\text{pos}}/w_{\text{ori}} = 100/4$, and the values for the scaling α and task stiffness k are set to default ($\alpha = 2, k = 35$ N/m).

Then, for each box, the operator must aim to toss on each target, in the sequence given by Figure 5. When the box is successfully tossed on the first target, the operator aims for the second target, and so on, until the box is tossed successfully on each target. The operator tries to do this in the least possible amount of attempts. A maximum number of attempts for each target is set at 10, if this amount is reached, the target will be identified as uncompleted. Each time a box is tossed successfully on a target, the amount of attempts is registered. The tossing result is called successful when it complies with the following three requirements: (1) the box ends up within the bounds of the target, (2) the short edge of the box is pointed perpendicular to the moving direction of the conveyor with an orientation accuracy of at least $30^\circ \pm 5^\circ$, and (3) the barcode is pointing up. In Figure 6, an example is shown of a successful tossing result and three unsuccessful results. This figure illustrates that the second, third and fourth toss are unsuccessful because of tossing requirements (1), (2), and (3) respectively. The parameter value which requires the least number of attempts to toss each box on the intended target, is chosen for that parameter.

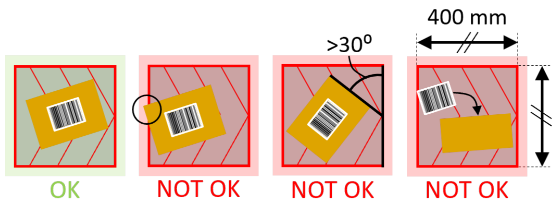


Figure 6: Examples to show if a tossing result is successful or not.

H.2 Results

Tables 4, 5, and 6, denote the amount of attempts that were needed to toss the box onto each target, as explained in the procedure, for each experiment set, see Appendix H.1.

During the experiments, we experienced that the teleoperation system is very suitable to demonstrate robotic tossing where boxes have to land in dedicated target areas. Tossing such that the boxes would make impact on particular locations was easy. Tossing boxes such that they end up close to the location where they made impact, however, was more challenging. Toss Box 3 and 4, proved to be far more challenging, as these would easily tumble after impact. Moreover, a higher QP stiffness gain gives the operator the feeling of having more control over the motion. For scaling, the operator personal preference laid on $\alpha = 2$ and $\alpha = 2.5$. The task weight ratio that the operator preferred lays between $w_x/w_\varphi = 100/12$ and $w_x/w_\varphi = 100/8$.

Based on the results in the tables, the chosen values for the controller's stiffness gain $k = 45$ N/m, position reference scaling $\alpha = 2$, and task weight ratio $w_x/w_\varphi = 100/10$, see Tables 4, 5, and 6, respectively.

Table 4: Number of attempts to toss each box in each target for different controller stiffness gains, an X counts for 10 attempts.

Set A	Number of attempts:									
Box:	Box 3			Box 4			Box 5			
Target:	1st	2nd	3th	1st	2nd	3th	1st	2nd	3th	*Totals:*
A-1: $k = 40$ N/m	4	5	7	2	2	5	2	3	1	31
A-2: $k = 30$ N/m	6	9	X	6	5	8	2	4	2	54
A-3: $k = 35$ N/m	2	6	9	1	2	5	1	2	2	30
A-4: $k = 25$ N/m	5	7	X	2	3	X	6	2	2	47
A-5: $k = 45$ N/m	3	4	5	2	1	6	1	1	2	**25**

Table 5: Number of attempts to toss each box in each target for different reference position scaling factors, an X counts for 10 attempts.

Set B	Number of attempts:									
Box:	Box 3			Box 4			Box 5			
Target:	1st	2nd	3th	1st	2nd	3th	1st	2nd	3th	*Totals:*
B-1: $\alpha = 3$	5	2	8	6	4	2	4	4	4	39
B-2: $\alpha = 1$	3	4	9	5	7	X	3	5	4	50
B-3: $\alpha = 2$	2	3	2	1	2	2	1	1	2	**16**
B-4: $\alpha = 1.5$	3	1	4	2	2	3	1	1	2	19
B-5: $\alpha = 2.5$	1	2	4	3	2	3	3	1	1	20

Table 6: Number of attempts to toss each box in each target for different QP task weight ratios.

Set C	Number of attempts:									
Box:	Box 3			Box 4			Box 5			
Target:	1st	2nd	3th	1st	2nd	3th	1st	2nd	3th	*Totals:*
C-1: $w_x/w_\varphi = 100/10$	1	2	1	1	3	1	1	3	3	**16**
C-2: $w_x/w_\varphi = 100/8$	1	3	3	1	3	2	1	1	2	17
C-3: $w_x/w_\varphi = 100/6$	2	4	3	2	3	4	1	1	2	22
C-4: $w_x/w_\varphi = 100/4$	3	3	4	2	4	4	2	1	2	25
C-5: $w_x/w_\varphi = 100/12$	2	3	4	1	2	4	2	3	2	23

I Statistics Related Concepts

I.1 Standard Deviation and Confidence Interval

To quantify how precise the result of a toss can be repeated, we use the standard deviation of to measure the amount of variation in a set of tossing outcomes [121]. To compute the (corrected) sample standard deviation of the rest-position of a box in $({}^U\mathbf{o}_B)_\mathbf{x}$ we use

$$\sigma_{\mathbf{x}_B} = \sqrt{\frac{\sum_{i=1}^{n}\left(({}^U\mathbf{o}_B)_{\mathbf{x},i} - ({}^U\bar{\mathbf{o}}_B)_{\mathbf{x},i}\right)}{n-1}}, \tag{I.1}$$

where $({}^U\mathbf{o}_B)_{\mathbf{x},i}$ denotes the value of $\mathbf{b}$, for the i^{th} toss, $({}^U\bar{\mathbf{o}}_B)_{\mathbf{x},i}$ denotes its mean, and n is the total number of tosses for each experiment.

Since the standard deviation is obtained by a limited number of samples, the value for the standard deviation that we compute using I.1, has an error margin. Using the confidence interval (CI), we can quantify an interval for which we can determine the computed standard deviation with a certainty of 95% [121]. For example, given a desired level of confidence, which is denoted by the Z-score, we can determine an interval within which the value for $\sigma_{\mathbf{x}_B}$ lays with a 95% certainty. We compute this by

$$\mathrm{CI}_{\mathbf{x}} = ({}^{U}\bar{\mathbf{o}}_{\mathbf{b}})_{\mathbf{x},i} \pm Z \times \frac{\sigma_{\mathbf{x}_B}}{\sqrt{n}}, \tag{I.2}$$

where Z denotes a standard Z-score for the desired level of confidence, and $\sigma_{\mathbf{x}}$ is the sample standard deviation (I.1). For the tossing results in this work, we choose to report them with a confidence level of 95%, which corresponds to a Z-score of 1.96.

I.2 Determine Appropriate Sample Size

To determine the appropriate sample size of tossing repetitions to determine the repeatability, a calculation is made. The appropriate sample size is based on the central limit theorem [121], where an estimation of the required sample size $\hat{n}$, can be calculated using

$$\hat{n} = \frac{4Z^2\hat{\sigma}^2}{W^2}, \tag{I.3}$$

where Z denotes a standard Z-score for the desired level of confidence, $\hat{\sigma}$ is an estimation of the expected standard deviation of tossing repetitions, and W is twice the margin of error that we accept for reporting the standard deviation. The estimated amount of required samples is calculated using a Z-score of 1.96, which corresponds to a 95% confidence interval of the estimation. Based on preliminary tests with tossing repetitions, the expected standard deviation between rest positions of the box $\left({}^{U}\mathbf{o}_B\right)_x$ and $\left({}^{U}\mathbf{o}_B\right)_y$ is estimated to be $\hat{\sigma} = 20$ mm. We choose the accepted margin of error of the calculated standard deviation to be between 5 and 10 mm. Based on these variables and the general rule that for using the central limit theorem the sample size must be $\geq$ 30 [122], the estimated number of required samples is chosen to be 30.

J Analysis of the Pick Pose Sensitivity

J.1 Permissible Pick Pose Deviation

Based on the sensitivity results for the short and long range toss, the limits on the pick pose deviation are explored. It is worth noting that the bounds on the pick poses are only investigated for two tossing trajectories and one box. To give an indication of the permissible variations in pick pose parameters, more research is required. However, by means of these experiments, we

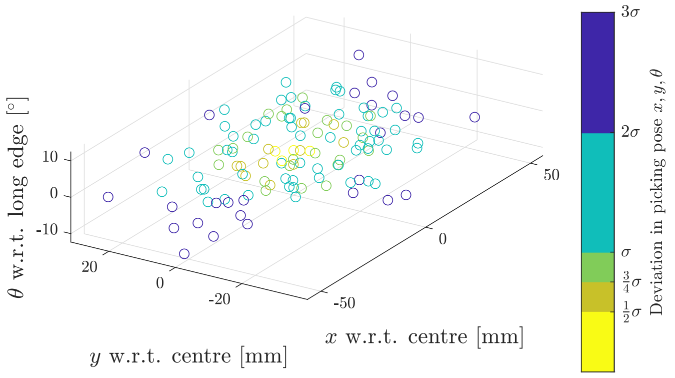
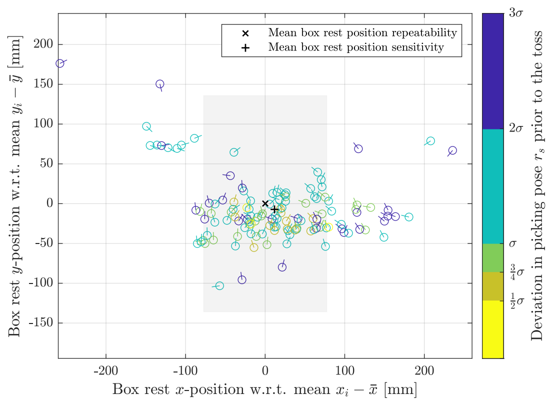
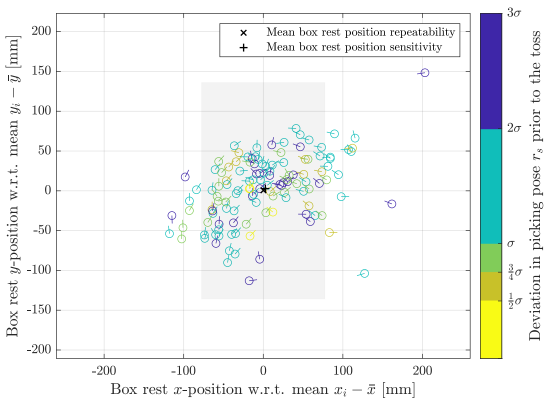


Figure 7: Division of pick pose samples according to the order of deviation in the pick pose **p**.



(a) **b** for the **short range** tosses.



(b) **b** for the **long range** tosses.

Figure 8: Box rest-poses **b** of the sensitivity experiments, with respect to the mean box rest pose from the repeatability experiments, for Gaussian distributed box pick poses. The color bar displays the box picking deviation. The rest position of each toss is the center of the circles, where the line gives a more intuitive indication of the rest-pose orientation.

can gain understanding of what could be admissible pick pose deviations. To find the permissible variations in the pick poses, the sampled pick poses are divided in five ranges. The first range contains the pick pose samples where all parameters have values $\leq \frac{1}{2}\sigma$, the second range includes all pick poses for which the deviations in all pick pose parameters are $\leq \frac{3}{4}\sigma$. The third, fourth, and fifth range include all samples where the pick pose parameters have deviations in the range $[0, \sigma]$, $[0, 2\sigma]$, and $[0, 3\sigma]$, respectively. The first range consists of only three samples. The second, third, fourth and fifth range consist of 15, 30, 98 and 125 samples, respectively. The division of pick pose samples, in each aforementioned range, is visualized in Figure 7.

To find the bounds on the pick pose deviations for which robotic tossing leads to predictable tossing results, the pick pose ranges are reflected on the box rest-poses after tossing. Figures 8a and 8b show the rest-pose after each toss i, related to the deviations in the pick pose, prior to each toss i. From the figures, a clear relationship between the pick pose deviation and tossing results can be observed. For higher deviations in the pick pose, as expected, the tossing result is clearly less precise, as these tosses often result in box' rest poses relatively far from the mean. Pick poses with deviations $> \sigma$ (denoted by the light and dark blue color in the figures) lead to tossing results with outliers far away from the mean box rest-pose $\bar{\mathbf{b}}$. For smaller deviations in the pick pose, where the deviation in the pick pose for all parameters is $\leq \sigma$, leads to more predictable results. Moreover, for tosses where the pick pose deviated in the range $[\frac{3}{4}\sigma, \sigma]$ in each parameter, the box' rest orientation shows large differences between tosses. Looking at all samples that had a pick pose deviation in the range $[0, \frac{3}{4}\sigma]$, the tossing results seem to have small deviations between tosses. In Tables 7 and 8, for the short range and long range tosses respectively, the deviation and precision of tosses for each range of deviations in the pick pose parameters are given. For reference, the repeatability results are also denoted in this table.

Perturbations in the pick pose introduce more deviation in the tossing outcomes, and the outcomes are less precise. However, for small perturbations in the pick pose, it looks like predictable tossing can still be achieved if the short range trajectory is repeated. Based on the results presented here, we believe that for appropriate tossing demonstrations, small deviations in the pick pose are admissible. During these experiments, as discussed in Section 4.4.2, the covariance for $\left({}^{B}\mathbf{o}_{S}\right)_{\mathbf{x}}$, $\left({}^{B}\mathbf{o}_{S}\right)_{\mathbf{y}}$, and $\boldsymbol{\varphi}$, are $3\sigma_{\mathrm{x_p}} = 75$ mm, $3\sigma_{\mathrm{y_p}} = 40$ mm, and $3\sigma_{\varphi_{\mathrm{p}}} = 30°$. Based on the results given in Table 7, it seems like the admissible pick pose deviation for appropriate tossing demonstrations is equal to a covariance of $\frac{3}{4}\sigma$ for each pick pose parameter. This would mean that the admissible range for the pick pose parameters is equal to $\left({}^{B}\mathbf{o}_S\right)_{\mathbf{x}} = [-6.67, 6.67]$ mm, $\left({}^{B}\mathbf{o}_S\right)_{\mathbf{y}} = [-12.5, 12.5]$ mm, and $\boldsymbol{\varphi} = [-5°, 5°]$. However, based on the analysis presented, we do not consider any scaling between how much each pick pose parameter affects the outcome of a toss. Therefore, it might be that the aforementioned range is not representative. To choose a more appropriate range for the required pick pose precision, we made an analysis to investigate how each pick pose parameter affects the outcome of a toss. If we know how much a certain value for a pick pose parameter affects the outcome of a toss, we can make an estimation of what is an appropriate scaling between the pick pose parameters. Based on this estimation, we can give a more accurate estimation of what could be admissible deviations in the pick pose, for predictable tossing. The analysis where we investigate the effect of each pick pose parameter on the outcome of a toss, is presented in Appendix J.2.

J.2 Picking Parameters Effect

We want to analyze the results of the sensitivity tossing outcomes, to find out how each pick pose parameter affects the box' rest-pose after tossing. The knowledge we gain here, can be used to decide a more appropriate scaling between the admissible pick pose precision of each parameter, as explained in Appendix J.1. As described in Section 4.4.2, during the sensitivity experiments, 125 pick poses were sampled from a Gaussian distribution. For each sampled pick pose, there are three (coupled) parameters, which all affect the outcome of a toss. To explore the effect of each parameter individually, the pick pose samples are divided into five ranges that all have the same

Table 7: **Short range** tossing results for different orders of deviation in the pick poses. In each column, the results are given for only the pick pose samples that have a deviation in the specified range (e.g. $[0, \frac{1}{2}\sigma]$). The deviation of the rest-pose is given with a confidence interval of 95%.

Toss Result:	**Unit:**	**Repeatability**	$[\mathbf{0}, \frac{1}{2}\sigma]$	$[\mathbf{0}, \frac{3}{4}\sigma]$	$[\mathbf{0}, \sigma]$	$[\mathbf{0}, \mathbf{2}\sigma]$	$[\mathbf{0}, \mathbf{3}\sigma]$
$\sigma_{\mathbf{x}_b}$	[mm]	17.7 ± 6.3	49.0 ± 48.0	37.5 ± 18.4	59.0 ± 18.5	69.1±13.7	81.0±14.2
$\sigma_{\mathbf{y}_b}$	[mm]	11.5 ± 4.1	22.7 ± 22.2	27.2 ± 13.3	27.7 ± 8.7	36.6±7.3	42.4±7.4
σ_{θ_b}	[°]	10.5 ± 3.7	26.2 ± 25.7	23.3 ± 11.4	40.6 ± 12.7	53.5±10.6	57.3±10
$\Delta\left({}^{U}\mathbf{o}_B\right)_{\mathbf{x}}$	[mm]	104.7	103.0	119.2	214.6	357.2	493.5
$\Delta\left({}^{U}\mathbf{o}_B\right)_{\mathbf{y}}$	[mm]	55.0	24.4	57.2	63.9	200.5	271.9
$\Delta\boldsymbol{\theta}$	[°]	44.8	58.9	75.9	136.7	335.3	335.3

Table 8: **Long range** tossing results for different orders of deviation in the pick poses. In each column, the results are given for only the pick pose samples that have a deviation in the specified range (e.g. $[0, \frac{1}{2}\sigma]$). The deviation of the rest-pose is given with a confidence interval of 95%.

Toss Result:	**Unit:**	**Repeatability**	$[\mathbf{0}, \frac{1}{2}\sigma]$	$[\mathbf{0}, \frac{3}{4}\sigma]$	$[\mathbf{0}, \sigma]$	$[\mathbf{0}, \mathbf{2}\sigma]$	$[\mathbf{0}, \mathbf{3}\sigma]$
$\sigma_{\mathbf{x}_b}$	[mm]	62.8 ± 22.5	15.9 ± 15.6	53.0 ± 26.0	55.7 ± 17.5	58.3±11.5	60.2±
$\sigma_{\mathbf{y}_b}$	[mm]	34.9 ± 12.5	31.4 ± 30.8	33.3 ± 16.3	31.1 ± 9.8	40.2±8.0	42.3±10.6
σ_{θ_b}	[°]	44.8 ± 16.0	37.1 ± 36.4	57.9 ± 28.4	54.5 ± 17.1	65.5±13.0	68.5±12.0
$\Delta\left({}^{U}\mathbf{o}_B\right)_{\mathbf{x}}$	[mm]	204.5	29.6	176.6	214.6	245.2	261.4
$\Delta\left({}^{U}\mathbf{o}_B\right)_{\mathbf{y}}$	[mm]	145.1	59.7	110.4	114.1	182.2	321.0
$\Delta\boldsymbol{\theta}$	[°]	134.8	95.1	180.9	180.9	354.6	354.6

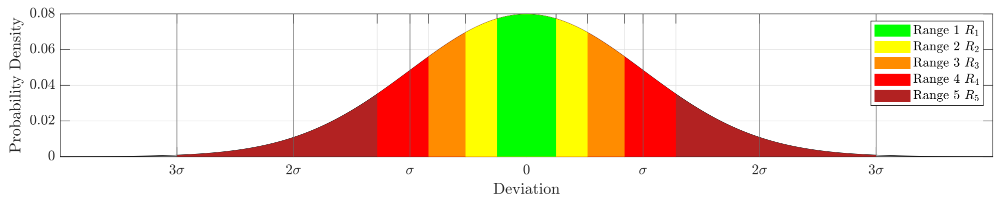
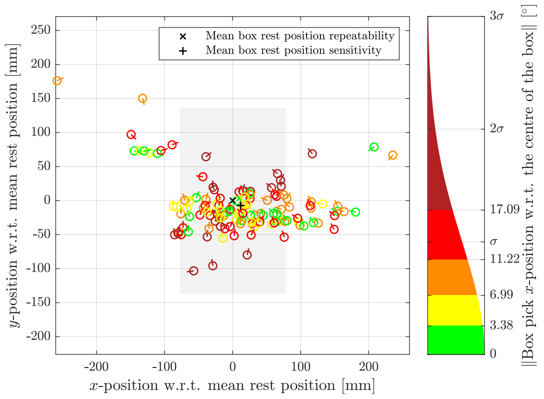


Figure 9: Gaussian distribution divided in five ranges that each have a 19.94% probability.

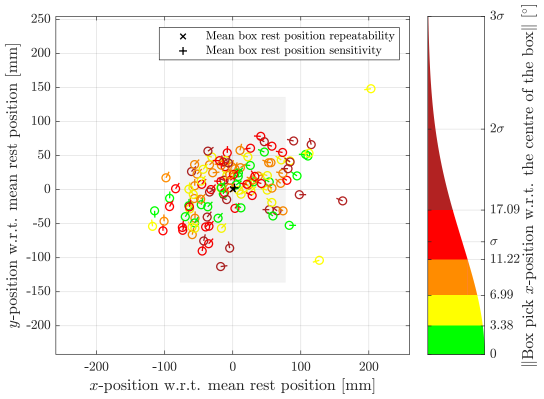
probability. By doing this, the pick pose distribution of 125 samples is divided such that each range (statistically) should contain 25 samples. This way, it is possible to divide the pick pose sample distribution in ranges by the order of deviation for one parameter, whiles the other two parameters remain Gaussian distributed. Figure 9 depicts a Gaussian distribution, which is divided into five areas: green ($\approx \pm[0, 0.254\sigma)$), yellow ($\approx \pm[0.254\sigma, 0.524\sigma)$), orange ($\approx \pm[0.524\sigma, 0.842\sigma)$), red ($\approx \pm[0.842, 1.282\sigma)$), and dark red ($\approx \pm[1.282, 3\sigma]$). Each for a certain parameter, the chance that a sample is taken from each of these areas is equal (there is a probability of 19.94% to take a sample from each of these areas, since the probability for picking in the range $[-3\sigma, 3\sigma]$ is 99.7%). Moreover, in each division of one parameter, the other two parameters remain Gaussian distributed. The division of the areas is calculated using the inverse cumulative distribution function for the three-parameter distribution [123]. For any sample within each of the areas, the other two parameters are Gaussian distributed, and thus it is possible to analyze how each pick pose parameter affects the outcome of a toss. Furthermore, each range of data samples is roughly comparable in size. The division as depicted in Figure 9, can be projected on each parameter. For example, when a parameter $\mathrm{x}^b_{\mathbf{p}}$ is analyzed, the tossing results can be shown for five ranges of deviations in parameter $\mathrm{x}^b_{\mathbf{p}}$. To analyze the effect of small and larger deviations in picking parameter x equally, the pick pose deviation is divided in ranges of $R_{1_x} = [0, 3.38)$, $R_{2_x} = [3.38, 6.99)$, $R_{3_x} = [6.99, 11.22)$, $R_{4_x} = [11.22, 17.09)$, and $R_{5_x} = [17.09, 40]$. Each of these ranges have the same probability of being sampled, and the other pick pose parameters $\mathrm{y}^b_{\mathrm{p}}$ and θ^b_{p} follow a Gaussian distribution in each range. It is worth noting that since this exploration is only done for 125 short and long range tosses, the number of samples might not be sufficiently large to base conclusions on or detect patterns at all. Therefore, the information presented below, is just a first exploration of how each pick parameter affects the box' rest-pose.

Pick Pose Position

First, the effect of deviations on the picking positions on the box is analyzed. Figures 10a and 10b depict the tossing results for the sensitivity study for the short and long range toss, respectively. In the graph, the color bar relates the deviation in the pick position in the x-direction, to the rest-pose of a box after tossing (see Figure 4.13). Table 9 and 10 list the standard deviations and precision of the rest-pose of the box, for each range of the picking positions in the x-direction R_{x1} to R_{x5} (see Figure 9), for the short and long range toss, respectively. Looking at the results in both the scatter plots as the standard deviations in the table, no clear effect of the pick position in the x-direction was observed. Therefore, it seems that the pick position in x-direction has limited contribution to the increased deviations in the rest-pose after tossing. In Figures 11a and 11b, the tossing results can be compared for different picking positions in the y-direction. Table 11 and



(a) 2D scatter plot of box' rest poses.



(b) 2D scatter plot of box' rest poses.

Figure 10: Box rest-poses **b** w.r.t. the mean, for Gaussian distributed box pick poses, for **short** (a) and **long** (b) **range** tossing. The color bar is divided into the pick pose deviation in in $\left({}^{B}\mathbf{o}_{S}\right)_{\mathbf{x}}$.

Table 12, show the standard deviation and precision of each rest-pose parameter after tossing with a picking positions in a range of the y-direction, for the short and long range toss, respectively. For the short range toss, it seems that the pick pose in y-direction affects the box' rest-pose after tossing in the x-direction. However, for the long range toss, this effect is not visible.

Pick Pose Orientation

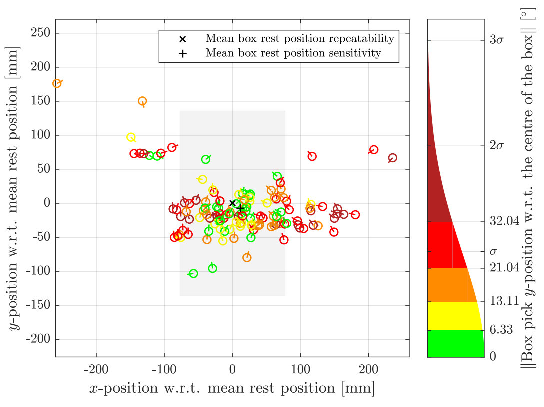
Finally, the effect of deviations on the picking orientation is analyzed. Figures 12a and 12b depict the tossing results for the sensitivity study, where the color bar displays the range of the picking orientation that a particular tossing repetition was tossed with. Table 13 and 14, give the standard deviation and tossing precision for each range of box picking orientations, for the short and long range toss, respectively. For both the long and short range tosses, a picking orientation of $\varphi < 2.62°$ leads to fairly smaller deviations in the box' rest-pose orientation. An even larger

Table 9: Deviation and precision the tossing results for five ranges of pick pose parameter $\left({}^{B}\mathbf{o}_{S}\right)_{\mathbf{x}}$, for the **short range** toss. The deviation of the rest-pose is given with a confidence interval of 95%.

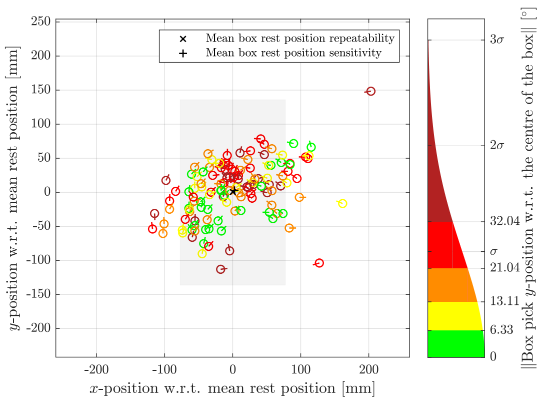
Toss Result:	**Unit:**	$\mathbf{SR_{x1}}$	$\mathbf{SR_{x2}}$	$\mathbf{SR_{x3}}$	$\mathbf{SR_{x4}}$	$\mathbf{SR_{x5}}$
$\sigma_{\mathbf{x}_b}$	[mm]	97.2 ± 40.6	61.7 ± 23.7	103.8 ± 38.4	71.6 ± 26.1	54.6 ± 31.8
$\sigma_{\mathbf{y}_b}$	[mm]	38.3 ± 16.0	30.7 ± 11.8	50.1 ± 18.6	39.9 ± 14.5	51.2 ± 23.0
σ_{θ_b}	[°]	72.3 ± 30.2	49.8 ± 19.1	68.5 ± 25.4	48.4 ± 17.6	38.4 ± 17.3
$\Delta\left({}^{U}\mathbf{o}_{B}\right)_{\mathbf{x}}$	[mm]	212.7	141.6	269.7	152.2	132.2
$\Delta\left({}^{U}\mathbf{o}_{B}\right)_{\mathbf{y}}$	[mm]	79.5	74.1	176.7	100.3	116.8
$\Delta\boldsymbol{\theta}$	[°]	237.5	123.6	221.9	135.8	100.7

Table 10: Deviation and precision the tossing results for five ranges of pick pose parameter $\left({}^{B}\mathbf{o}_{S}\right)_{\mathbf{x}}$, for the **long range** toss. The deviation of the rest-pose is given with a confidence interval of 95%.

Toss Result:	**Unit:**	$\mathbf{LR_{x1}}$	$\mathbf{LR_{x2}}$	$\mathbf{LR_{x3}}$	$\mathbf{LR_{x4}}$	$\mathbf{LR_{x5}}$
$\sigma_{\mathbf{x}_b}$	[mm]	62.0 ± 25.9	70.9 ± 27.3	51.4 ± 19.0	50.1 ± 18.2	67.2 ± 30.2
$\sigma_{\mathbf{y}_b}$	[mm]	32.8 ± 13.7	48.6 ± 18.7	31.3 ± 11.6	44.4 ± 16.2	52.1 ± 23.4
σ_{θ_b}	[°]	61.4 ± 25.7	88.7 ± 34.1	53.8 ± 19.9	70.7 ± 25.7	59.8 ± 26.9
$\Delta\left({}^{U}\mathbf{o}_{B}\right)_{\mathbf{x}}$	[mm]	119.9	211.8	102.4	103.5	166.1
$\Delta\left({}^{U}\mathbf{o}_{B}\right)_{\mathbf{y}}$	[mm]	58.2	149.0	68.4	90.4	116.5
$\Delta\boldsymbol{\theta}$	[°]	179.2	268.4	84.6	262.5	100.4



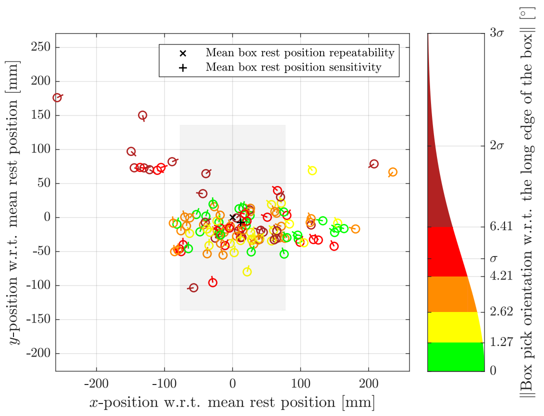
(a) 2D scatter plot of tossing results.



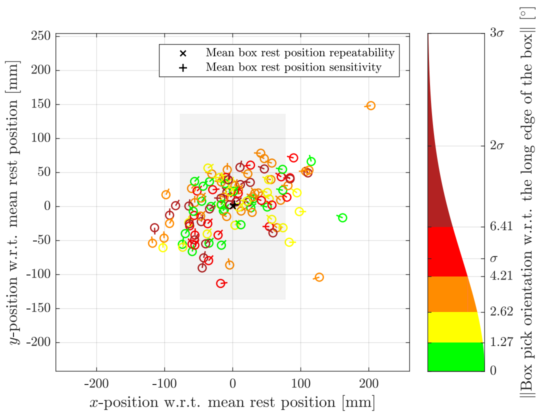
(b) 2D scatter plot of tossing results.

Figure 11: Box rest-poses **b** w.r.t. the mean, for Gaussian distributed box pick poses, for **short** (a) and **long** (b) **range** tossing. The color bar is divided into the pick pose deviation in $\left({}^{B}\mathbf{o}_{S}\right)_{\mathbf{y}}$.

deviation in the pick orientation does not clearly affect the box' rest position.



(a) 2D scatter plot of tossing results.



(b) 2D scatter plot of tossing results.

Figure 12: Box rest-poses (or, tossing results) w.r.t. the mean after the **short and long range tosses**, for Gaussian distributed box pick poses. The color bar is divided into the pick pose deviation in in $\boldsymbol{\varphi}$.

Conclusion on Pick Pose Parameter Analysis

In Appendix J.1, the results indicated the admissible range for the pick pose parameters is equal to $\left({}^{B}\mathbf{o}_{S}\right)_{\mathbf{x}} = [-6.67, 6.67]$ mm, $\left({}^{B}\mathbf{o}_{S}\right)_{\mathbf{x}} = [-12.5, 12.5]$ mm, and $\boldsymbol{\varphi} = [-5, 5]$ °. However, here, we did not include the effect of each pick pose parameter separately. Based on the results presented in this appendix, we can decide on a scaling of the pick pose deviations, according to how much each pick pose parameter affects the outcome of a toss.

From the results of the analysis presented in this appendix, we can not identify a clear effect on

Table 11: Deviation and precision of the tossing results for five ranges of pick pose parameter $\left({}^{B}\mathbf{o}_{S}\right)_{\mathbf{y}}$, for the **short range** toss. The deviation of the rest-pose is given with a confidence interval of 95%.

Toss Result:	**Unit:**	$\mathbf{SR_{y1}}$	$\mathbf{SR_{y2}}$	$\mathbf{SR_{y3}}$	$\mathbf{SR_{y4}}$	$\mathbf{SR_{y5}}$
$\sigma_{\mathbf{x}_b}$	[mm]	48.6 ± 18.7	53.1 ± 23.9	79.1 ± 79.1	94.0 ± 32.6	111.2 ± 48.7
$\sigma_{\mathbf{y}_b}$	[mm]	41.9 ± 16.1	37.0 ± 16.6	53.7 ± 20.3	41.6 ± 14.4	30.1 ± 13.2
σ_{θ_b}	[°]	52.0 ± 20.0	52.3 ± 23.5	49.1 ± 18.5	64.8 ± 22.5	65.5 ± 28.7
$\Delta\left({}^{U}\mathbf{o}_{B}\right)_{\mathbf{x}}$	[mm]	126.8	152.7	260.3	217.7	244.3
$\Delta\left({}^{U}\mathbf{o}_{B}\right)_{\mathbf{y}}$	[mm]	103.7	97.7	178.2	82.6	73.7
$\Delta\boldsymbol{\theta}$	[°]	123.1	136.2	168.6	237.5	221.9

Table 12: Deviation and precision of the tossing results for five ranges of pick pose parameter $\left({}^{B}\mathbf{o}_{S}\right)_{\mathbf{y}}$, for the **long range** toss. The deviation of the rest-pose is given with a confidence interval of 95%.

Toss Result:	**Unit:**	$\mathbf{LR_{y1}}$	$\mathbf{LR_{y2}}$	$\mathbf{SR_{y3}}$	$\mathbf{SR_{y4}}$	$\mathbf{SR_{y5}}$
$\sigma_{\mathbf{x}_b}$	[mm]	55.4 ± 21.3	66.5 ± 29.9	57.9 ± 21.8	59.1 ± 20.5	64.6 ± 28.3
$\sigma_{\mathbf{y}_b}$	[mm]	40.5 ± 15.6	39.0 ± 17.5	35.0 ± 13.2	43.2 ± 15.0	53.2 ± 23.3
σ_{θ_b}	[°]	47.0 ± 18.1	58.7 ± 26.4	56.1 ± 21.2	84.8 ± 29.4	84.2 ± 36.9
$\Delta\left({}^{U}\mathbf{o}_{B}\right)_{\mathbf{x}}$	[mm]	127.4	166.7	105.8	128.4	207.7
$\Delta\left({}^{U}\mathbf{o}_{B}\right)_{\mathbf{y}}$	[mm]	76.1	93.7	64.8	104.0	149.0
$\Delta\boldsymbol{\theta}$	[°]	90.6	91.0	90.4	263.6	263.5

Table 13: Deviation and precision of the tossing results for five ranges of pick pose parameter $\boldsymbol{\varphi}$, for the **short range** toss. The deviation of the rest-pose is given with a confidence interval of 95%.

Toss Result:	**Unit:**	$\mathbf{SR}_{\varphi\mathbf{1}}$	$\mathbf{SR}_{\varphi\mathbf{2}}$	$\mathbf{SR}_{\varphi\mathbf{3}}$	$\mathbf{SR}_{\varphi\mathbf{4}}$	$\mathbf{SR}_{\varphi\mathbf{5}}$
σ_{x_b}	[mm]	78.8 ± 31.5	63.1 ± 27.0	76.7 ± 26.6	78.0 ± 32.6	101.1 ± 39.6
$\sigma_{\mathbf{y}_b}$	[mm]	24.0 ± 9.6	35.8 ± 15.3	28.3 ± 9.8	43.5 ± 18.2	67.9 ± 26.6
σ_{θ_b}	[°]	38.4 ± 15.4	29.1 ± 12.4	51.8 ± 17.9	59.5 ± 24.9	87.2 ± 34.2
$\Delta\left({}^{U}\mathbf{o}_B\right)_{\mathbf{x}}$	[mm]	173.2	156.8	239.0	154.0	263.7
$\Delta\left({}^{U}\mathbf{o}_B\right)_{\mathbf{y}}$	[mm]	55.7	84.9	67.4	96.2	176.8
$\boldsymbol{\Delta\theta}$	[°]	80.3	51.3	221.9	110.3	251.6

Table 14: Deviation and precision of the tossing results for five ranges of pick pose parameter $\boldsymbol{\varphi}$, for the **long range** toss. The deviation of the rest-pose is given with a confidence interval of 95%.

Toss Result:	**Unit:**	$\mathbf{LR}_{\varphi\mathbf{1}}$	$\mathbf{LR}_{\varphi\mathbf{2}}$	$\mathbf{LR}_{\varphi\mathbf{3}}$	$\mathbf{LR}_{\varphi\mathbf{4}}$	$\mathbf{LR}_{\varphi\mathbf{5}}$
σ_{x_b}	[mm]	58.0 ± 23.2	56.4 ± 24.1	71.3 ± 24.7	51.9 ± 21.7	56.2 ± 22.0
σ_{y_b}	[mm]	34.5 ± 13.8	34.6 ± 14.8	50.3 ± 17.4	46.6 ± 19.5	39.8 ± 15.6
σ_{θ_b}	[°]	47.3 ± 18.9	55.4 ± 23.7	85.4 ± 29.6	80.5 ± 33.6	58.7 ± 23.0
$\Delta\left({}^{U}\mathbf{o}_B\right)_{\mathbf{x}}$	[mm]	162.6	104.0	206.3	93.0	119.2
$\Delta\left({}^{U}\mathbf{o}_B\right)_{\mathbf{y}}$	[mm]	66.4	61.3	154.2	115.3	90.9
$\boldsymbol{\Delta\theta}$	[°]	83.4	90.4	261.5	264.3	179.4

the tossing outcomes that is related to the pick position $\left({}^{B}\mathbf{o}_S\right)_{\mathbf{x}}$ and $\left({}^{B}\mathbf{o}_S\right)_{\mathbf{y}}$. In the analysis of how the pick orientation φ affects the tossing outcomes. However, we observe that for the short range tosses, the outliers in the box' rest-poses were all picked with a large deviation in the pick-orientation φ. In the long range tossing outcomes, this relation is not clearly visible. To confirm if the chosen precision in the pick-orientation, indeed affects box' rest-poses more than the pick-position, there are more experiments required. Since the effect is already clearly visible for the short range tosses, we think that a more appropriate scaling must be chosen between the pick positions and orientations in the pick pose samples. If more sensitivity experiments are performed, we advise to increase the precision in pick orientation, and we think that the precision of the pick position can be relaxed. To this end, we propose that in a new set of experiments, the admissible range for the pick pose parameters is chosen to be $\left({}^{B}\mathbf{o}_S\right)_{\mathbf{x}} = [-10, 10]$ mm, $\left({}^{B}\mathbf{o}_S\right)_{\mathbf{x}} = [-15, 15]$ mm, and $\boldsymbol{\varphi} = [-2.5, 2.5]$ °.

To determine the effect of each separate parameter, it might be useful to perform experiments in which each parameter is deviated independently. In this case, however, it must be taken into account that the independent effect of deviations in the pick parameters does not certainly have the same effect on the tossing results as when the pick parameters are combined.

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Declaration Concerning the TU/e Code of Scientific Conduct for the Master's/PDEng/PhD Thesis

I have read the TU/e Code of Scientific Conduct[i].

I hereby declare that my Master's/PDEng/PhD-thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date:	26.04.2022
Name:	T.I.C. van Gorp
ID-Number:	1386573

Signature: [illegible]

Submit the signed declaration to csa.ieis@tue.nl

[i] See: https://www.tue.nl/en/our-university/about-the-university/integrity/scientific-integrity/
The Netherlands Code of Conduct for Academic Practice of the VSNU can also be found here. More information about scientific integrity is published on the websites of TU/e and VSNU.