DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Coupling of Simulink Controller and Robot Simulation for Simulating Compliant Contacts with the Environment

Niklas Lohmann

DEPARTMENT OF INFORMATICS

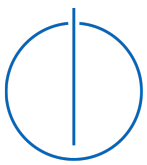
TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Coupling of Simulink Controller and Robot Simulation for Simulating Compliant Contacts with the Environment

Kopplung eines Simulink Reglers mit einer Robotersimulation zur Simulation nachgiebiger Umgebungskontakte

Author:	Niklas Lohmann
Supervisor:	Prof. Alin Albu-Schäffer
Advisor:	Adrian Simon Bauer
Submission Date:	15.4.2022



I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.4.2022 Niklas Lohmann

Acknowledgments

I wish to express my sincere thanks to my advisor, Adrian Bauer, for the continuous support and constructive advice at every point of my work at DLR.
I am also grateful to Prof. Alin Albu-Schäffer, Director of the RM institute, for providing a great scientific environment and pushing the Rollin' Justin project, which was the basis for my research.
I would also like to thank Dr. Daniel Leidner for the trust he placed in me for my work at DLR and for the continuous support throughout the work on my thesis, and Xuwei Wu for his support and his reliable answers to every single question regarding Simulink and the robot controller.
I also place on record, my gratitude to all members of the Rollin' Justin team and the FUTURO group for their encouragement, their advice and the great work atmosphere they provided even in times of social distancing.

Abstract

Humanoid robots often have a larger spectrum of abilities and requirements compared to specialized industry robots and interact with their environment in different and more complex ways. This comes with challenges when implementing and testing their behavior. Digital representations of these robots are often used to plan or visualize behavior but rarely to physically simulate movement as a way to develop a digital twin that also behaves identically in any scenario given the same input.
This thesis addresses the coupling of the existing controller of DLR's humanoid robot Rollin' Justin with a physics simulation, allowing for the reproduction, prediction and measurement of the robot's torque-controlled behavior and its contact forces to the environment. The compliant controller reactions to these contacts are crucial to the success of many robotic tasks.
The fully physics-driven simulation also makes it possible to test scenarios in advance and to make behavior predictions of the real robot, even in cases of uncertainty regarding the surrounding. Shifting the testing of scenarios or controller changes to the simulation can avoid the time and financial efforts of executions on the physical robot, which possibly even damage parts of it.
We improved the robot model used in the simulation, defined communication interfaces between the controller and the simulation and implemented the actuation logic, converting the pre-existing kinematic simulation behavior to a torque-driven one.
For the evaluation of the coupled system we measured the tracking accuracy of the simulated model in regard to recorded data from the real robot and conducted an experiment with environment contacts.
The digital twin shows good tracking accuracy, joint and end effector positions are replicated well. Forces in contact with the environment are at similar magnitudes as observed in reality. The experiment also demonstrates the usage of multiple simulated environment states to predict outcomes in cases of uncertainty.

Contents

1 Introduction

Humanoid robots like DLR's "Rollin' Justin" can be used for a large variety of tasks, especially ones normally executed by humans. They should integrate into more complex environments than industrial production robots and fulfil more complex tasks. Rollin' Justin is mainly used to explore possibilities and challenges of humanoid robots in extraterrestrial usage, e.g. in the previous Meteron Supvis-Justin experiment [1].
The accuracy and robustness requirements of Justin are significant for instance when connecting to a charger socket, since any error can lead to damage to its environment or the robot itself. Additionally, the environment is much more complex and susceptible to errors or inaccuracies than for instance a production conveyor belt scenario.

A simulation of Rollin' Justin and its environment was previously created to support the correct execution of these tasks. Justin is represented as a mechanical model that follows certain trajectories with its joints, either calculated directly or recorded on the real robot. This leads to multiple usages, visualized in yellow in Fig. 1.1:
The kinematic reproduction of a planned trajectory on the robot model can detect undesired collisions with the environment on the path and discard that trajectory on the real robot.
Additionally, the state of the world in the simulation can be evaluated and interpreted to discrete conclusions - a process called symbolic inference - much easier than analyzing those properties through image processing, for example determining whether a bottle is upright or lying on a surface.
Another useful feature of the existing simulation is its prediction abilities - testing the success of some operation by playing the behavior in the simulation and evaluating the world state afterwards, even with non-kinematic objects that are influenced by gravity and contact forces like dropping a ball into a container. This can prevent the robot from taking unsuccessful paths or indicate good success chances, even in realtime scenarios [2].

However, simulating a kinematic model of Justin that follows paths by directly setting its joint angles leaves out the dynamic behavior of the real robot. Forces and torques can not be measured in a simulation like this and the controller response is not simulated at all.

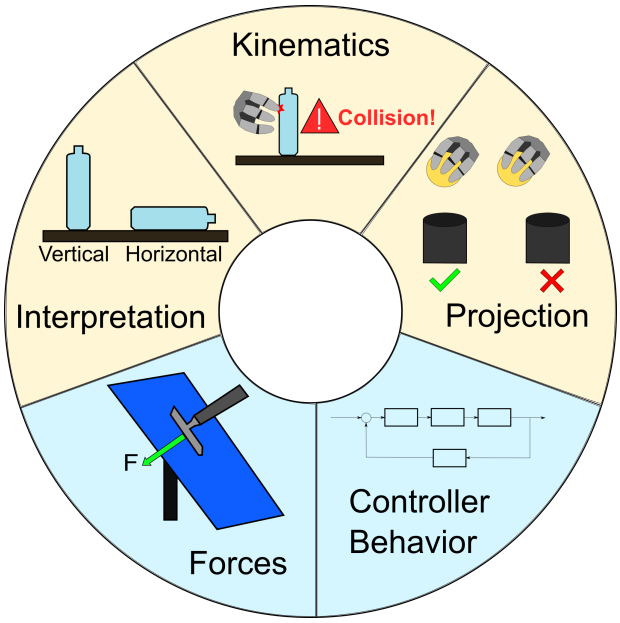


Figure 1.1: Previous (yellow) and new (light blue) capabilities of the simulated Justin

Predictions are only accurate if the movements do not contain contacts with the environment, because the compliant contacts that Justin's controllers produce are not replicated in the simulation.
In an application that involves contacts with a static environment like wiping the surface of a solar panel, the existing simulation would only detect there was a collision and move the tool into the panel leading to undefined behavior and unrealistic forces. Since the simulation only follows the paths interpolated as the commanded positions, it also doesn't reproduce the delay with which the real joints follow these positions through the controllers. This can only be countered by recording the joint position over time on the real robot, increasing the effort significantly and negating the predictive capabilities of a simulation.

This thesis is dedicated to the efforts to counter these problems by integrating the existing robot controller into the simulation to build a fully functional digital twin that represents its physical counterpart as accurately as possible.
Instead of setting joint positions directly which bypasses the physics simulation the robot is a fully dynamic and physics-driven construction composed of links and actuated joints. 'Links' refer to the physical bodies with inertial properties that are connected by hinge joints (also referred to as revolute joints).

The simulation and controller communicate with each other and synchronize their discrete time steps through blocking mechanisms. We implement all the different controller modes and improve the robot model in the simulation to reproduce the real behavior more closely.
By keeping the exact same controller and controlling the simulated model through torques the simulation reproduces trajectories on its own. It can also replicate the acting forces and torques realistically, including the controller reactions to the simulated forces and torques. This allows for the simulation and evaluation of compliant contacts, a core feature of Justin's behavior. These new functionalities are displayed in light blue in Fig. 1.1.
The digital twin can then ideally be used to simulate any scenario the real robot could execute and allows for the prediction and analysis of their outcomes. The simulation can also be used as a testing ground for changes to the controller.
Additionally, the symbolic inference functionality is extended to include forces and torques. Predictions can also include contact forces which can be interesting for the mentioned scenarios with desired forces, like Justin wiping the surface of a solar cell.
These capabilities save time, effort and costs since the preparation and execution of simulated scenarios is much easier and faster. Any commanded or planned behavior resulting in undesired outcomes can be avoided on the real robot, while failures in the simulation do not have any negative consequences.
The estimations of the world state are produced through image processing and are subject to uncertainties e.g. the position of a solar panel relative to Justin. The simulation can deal with inaccurate world representations by running the same scenario with small variations in the initial world state and comparing the results, an example scenario with two varying parameters is evaluated in this thesis. Live predictive usage of this functionality is quite conceivable by running simulations in parallel and deriving optimal approaches for the real robot's actions.

Chapter 2 will present a short overview over some related concepts and work to give a larger picture of simulations of non-industry robots.
Chapter 3 will go into detail about the systems used for Rollin' Justin and its simulation. The general coupling concept and its realisation in the form of two synchronized processes is explained in more depth together with an overview of the communication between simulation and controller. The different controller types implemented for the simulation and their respective use cases are presented. Some changes were made to the previous model of the robot in order to resemble the real robot more closely and to function in a torque-driven way.
Following that, in chapter 4 the tracking experiments on the digital twin are described regarding their setup, their evaluation and their results. The replication of joint posi-

tions and torques is crucial to the accuracy of compliant contacts.
We also evaluate a scenario of a wiping task with compliant environment contacts including two objects under pose uncertainty.
In chapter 5 the results are discussed in more detail, the use cases and boundaries of the created system are defined and further improvements to the model are explored.
Chapter 6 draws a conclusion on the achievements and expectations of the coupled systems and a short outlook on further improvements and usages of the developments is presented.

2 Related Work

In the following the humanoid robot Rollin' Justin and its use cases are explained, an overview of dynamic robot simulations is given and the term 'digital twin' is introduced.

2.1 Rollin' Justin

The humanoid robot Rollin' Justin at DLR is built for executing human tasks while being controlled remotely, even from the international space station. A large number of these tasks involves contacts to movable objects or the environment with quite different implications, for instance whether a surface should be penetrated, or whether a deformation takes place. This leads to a classification on a higher level to allow for different modes of operation depending on the task [3]. The ability to interact with objects in different ways reliably and with a significant error tolerance is crucial to the use of the robot.
To accommodate for this multitude of use cases, different controllers are available for the movements and actions, a state feedback controller for maximum accuracy - useful for picking and placement of objects - and two impedance controllers for compliant behavior in contacts. These allow for the exertion of controlled moderate forces on the environment or unexpected obstacles, which is crucial for interactions with humans and careful manipulation of objects like wiping dust from surfaces or cleaning the floor. By setting the commanded end effector positions to a point where they would penetrate a surface with a certain depth, a desired force will be applied by the controller once the end effector reaches the surface.
The compliant contacts can also integrate more sophisticated approaches to environment interactions. The inference of a surface contact can be realized through a combination of the measured external force and the position of the hand or tool to enable purely haptic feedback to drive movement corrections or success rating instead of visual feedback that may not be available in the required quality e.g. for wiping particles from surfaces [4].

2.2 Dynamic Robot Simulations

Static models or simulations of robots are used for a variety of reasons, granting a 3D representation of robot parts and their states, planning movements in an environment or interpreting the world information. Dynamic robot simulations offer a fully physically simulated world that extends beyond kinematic analysis by controlling the robot and the environment through motor torques and forces. This concept also requires the integration of a controller to command the required torques.

2.2.1 Reality Gap

The reality gap refers to the discrepancies between an observed behavior in reality and a digitally simulated and expected one given an identical input and initial state. This difficulty of transferring knowledge can frequently be observed in the context of robot simulations and their physical counterparts, independent of the direction of the transfer [5]–[7].
With the higher demands regarding robots' abilities, reliability and compatibility with humans, the importance of this reality gap grows. Making assumptions and inferences about a robot's environment is crucial to its locomotion and interactions but e.g. preemptively checking for collisions along a movement path requires some digital representation of the robot which is prone to the reality gap.
An approach gaining popularity for tackling complex robot structures and locomotion problems is training the controller parameters inside of a simulation and then using the same controller on the real robot [8]. This procedure is often called "Sim-To-Real transfer" and is subject to the reality gap, leading to inaccurate or even unexpected behavior once the controller is transferred to reality. Multiple approaches are available to train behavior and to optimize the transferability of the trained parameters to the real system and minimizing the reality gap, for instance domain randomization - introducing noise to sensor data - or integrating analytical knowledge [9]. However a generally optimal approach is not yet found [10].
The reality gap can occur for multiple reasons like the model being simplified or inaccurate or making incorrect assumptions about the environment. The system presented in this thesis does not use learning algorithms for its controllers, but it faces the same difficulties of creating a simulation as robust and accurate as possible to recreate real conditions. It still offers the possibility to implement learning algorithms in the future or could be used to test changes to the controller in a safe and quick way.

2.2.2 Usages

Advanced legged robots often require a large number of joints and therefore degrees of freedom (DOF). They especially profit from simulations by testing controller behavior. This is the case for the BigDog robot at Boston Dynamics [11] which can navigate steep and difficult terrain.
The ANYmal robot at ETH Zürich is also a quadruped robot and successfully learned its controller parameters through simulation [8]. Boston Dynamics' humanoid Atlas robot was even subject to a contest of simulation training with multiple challenges [12]. Other robot simulations focus on different challenges, like robot interactions with soft bodies [13] or compliant contact behavior [14], which is also a focus of this thesis.

2.3 Digital Twin

The term 'digital twin' is currently used in different contexts, a general definition of the usage and meaning of the term is provided in [15, p. 15]:

> A set of adaptive models that emulate the behaviour of a physical system in a virtual system getting real time data to update itself along its life cycle. The digital twin replicates the physical system to predict failures and opportunities for changing, to prescribe real time actions for optimizing and/or mitigating unexpected events observing and evaluating the operating profile system

Digital twins are commonly used in industrial domains as a representation of a production process including products, parts, robots or other machines, fulfilling most or all of the purposes of the given definition from live visualization to higher-level production planning and prediction. The application areas are therefor quite broad ranging from manufacturing [16] and construction [17] to aviation [18]–[21] and medical precision applications [22].
The system developed in this thesis fulfils most of these requirements and could easily be adapted to run with live input from the real robot.
Digital twins in general are usually subject to the reality gap and the efforts to improve a DT are aimed at minimizing this gap. Depending on the application domain, the reality gap can cause significant problems e.g. in aviation.
In the following the term digital twin will refer to a static or dynamic digital representation of a robotic system, which includes the previously existing kinematic simulation of Rollin' Justin and the simulations mentioned in section 2.2.2.

3 Coupling of Controller and Simulation

In order to simulate the dynamic robot behavior in the same way Rollin' Justin behaves in reality, the controller commands the joint 'motor' torques in the simulation instead of the real world. The simulation also contains the world around the robot to resemble reality as closely as possible and allow for environment interactions and collisions.
There are two obvious solutions to couple the simulation with a controller: Implementing a new controller that is mimicking the real controller as closely as possible directly inside the simulation software or running the actual controller and replacing its input and output interfaces with ones that allow for communication with the simulation software.
The second approach comes with the downsides of increased effort for communication and higher error susceptibility and the upsides of reaching extremely high replication accuracy, reusage of existing software and much easier adaptation to changes in the Rollin' Justin controller.
We chose the second approach in order to provide a digital twin of the robot that reproduces the behavior of the physical robot as accurately as possible given the same input.

3.1 Systems Used

The existing controller is implemented in Matlab Simulink. We added blocks for the exchange of data with the simulation. Furthermore, expected input from the robot hardware was replaced by constants.
The simulation environment Gazebo is used with custom plugins responsible for the specific communication requirements and for controlling the digital twin of Rollin' Justin.
As a central structure, one process called "Links and Nodes" - in short LN - starts all the required sub-processes. In our case, it mainly serves as the communication interface between Simulink and Gazebo.
LN starts both processes, monitors them and provides named data buffers that can be written to and read from. Both Simulink and Gazebo use a library to exchange data with these buffers. LN also offers an integrated python scripting interface to interact

with the processes without the setup of custom processes. This is used to set the desired paths or poses the robot should move to.
Apart from the communication blocks and constants the Simulink controller remains unchanged from the one used on the real robot.
The simulation recreates the motors and sensors of the real robot with simulated joints whose mechanical properties aim to replicate those of the real joints. Section 3.6 goes into more detail on the improvements of the robot model to resemble real constructions more closely.

3.2 General Concept

Movements are executed by the robot by applying the torques calculated by the controller according to the current robot state. This sentence already indicates that a typical closed control loop with a set point, controller, process variable and measurement needs to be realized.
The Simulink controller takes over the obvious task, the simulated joints form both the final control element and the process (since they are actuated directly without motors in the simulation) and the simulation also offers exact measurements of the joint positions. The terms of joint 'positions' and 'angles' are used interchangeably.
The data flow in a normal execution is shown as an overview in figure 3.1 and consists of multiple steps:

- Through some user interaction one or more desired end positions are sent to Simulink
- An interpolation from the current pose to the desired pose is calculated continuously over the following timesteps, which updates the set point - desired joint angles - inside the controller and therefore produces a continuous change in torques.
- Repeatedly, the Simulink controller receives the current robot pose from the simulation and calculates the errors of the joint angles and from this the torques that should be applied to the joints.
- Simulink sends the torques to Gazebo, where they are applied, the robot pose updates, gets measured and sent back to Simulink for the next timestep.

This alternating execution is looped at 1kHz in simulation time to approximate a continuous behavior just like on the real robot. Notably, the simulation does not need to

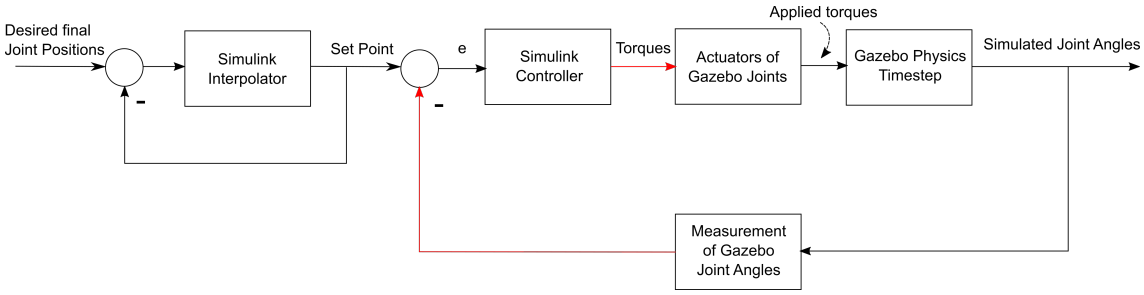


Figure 3.1: Control loop communicating between user, simulation and controller Simulation - Controller interfaces marked in red

run in realtime and keeps simulating every step with a time difference of 1ms without regard to the real time passed between two steps.

3.3 Data Interfaces

Table 3.1: Data from Gazebo to Simulink

Parameter	Description	Data type / Unit
have_command	Only used in the physical robot for synchronization.	uint8
telemetry_counter	Counts the exchanged packets and guarantees a difference between two following packets.	uint8
state	Only used in the physical robot for status information about every joint (power, emergency break)	double[]
motor_pos+poti_pos	Joint angles measured by separate sensors on the real Justin, treated the same in the DT.	double[], in Radians
torque	Measured torques on the joints. Can be used by Simulink for collision detection.	double[], in Nm
datacontainer	Only used in the physical robot for logging.	double[]

Table 3.2: Data from Simulink to Gazebo

Parameter	Description	Data type / Unit
request_command	Only used in the physical robot for synchronization.	uint8
command_counter	Counts the exchanged packets and guarantees a difference between two following packets.	uint8
control	Indicates whether a joint is controlled by torque or by its own low-level position control. 1 for force control, 0 and 4 for position control.	uint8[]
desired_pos	The current desired angle for each joint.	double[], in Radians
torque_in	Required torques on the joints.	double[], in Nm
speed	Unused parameter, always zero.	double[]
zr_kp	P gain of the motor position feedback (used in simulation for state space control).	double[]
zr_kd, zr_c2	Unused: D and I gain of the motor position feedback	2x double[]
zr_kt, zr_ks, zr_c1	Unused: P and D gain of the motor torque feedback and cutoff frequency of the low-pass filter for the motor torque feedback	3x double[]

The main task of system coupling in general lies in the correct definition and use of interfaces, i.e. the internal functioning of the programs should remain unchanged where possible.
The main components of the control loop are already implemented: The Simulink interpolator and controller as well as the Gazebo physics simulation and API. The main data interfaces used to couple Gazebo and Simulink are also visible in the control loop in red (Fig. 3.1): The segment between the controller and Gazebo actuators transports the torques (along with other data) and the joint position measurements from Gazebo return to the controller to determine the error term.
The data buffers in LN are also referred to as topics and are classified by the contained set of data types. Topics offer a publish-subscribe pattern to provide asynchronous communication capabilities. Data exchange therefore consists of two parts: The data

publisher writes its data into a topic of the LN process, the subscriber read call gets unblocked and copies the data into a local buffer for further usage.
The specific data exchange required between Simulink and Gazebo is listed in table 3.1 and 3.2. Specific focus lies on the exchange of torques and joint positions. The controller send required torques and the current desired pose, the simulation in turn sends the simulated pose and measured torques back to Simulink, including effects of contact forces with other objects.
Notably the last 5 parameters sent by Simulink are not used for simulation purposes, they are used for the fourth and fifth order state feedback controllers realized on the real robot on each joint [23], this controller will be referred to as the state space controller. It is realized on the simulated robot by a PD controller with an empirically determined D gain and the transmitted *zr_kp* as the P gain.
Simulink additionally has a data interface for receiving the desired poses from the user. These poses go through a quadratic spline interpolator and get transferred into the controller as desired angles of the joints. For the user, this interface is encapsulated in a python script which is responsible for the formatting and transmission of the poses the user enters by providing desired angles for each joint.

3.4 Synchronization

A big difference between using the controller in the digital twin and the real robot is that the Gazebo simulation also processes in discrete timesteps while the motors and sensors work continuously in reality.
This means that while running the controller and looking up the sensor values every millisecond does work for the real robot, in the digital twin this leads to synchronisation issues: If the controller or the simulation don't run in realtime for performance reasons or due to uneven OS scheduling, one of the processes will use the same input data twice or skip data from the other process, leading to incorrect behavior that wouldn't occur on the real robot.
Instead, every timestep of the simulation is calculated as one millisecond and must be followed by one timestep of the controller also calculated as one millisecond, then another timestep of the simulation and so on. 'Calculating as one millisecond' does not imply a requirement on how long the calculation actually takes in reality. Since the two processes need to wait for each other, a blocking mechanism was implemented.

Synchronization is achieved on the side of Simulink using a custom library block that was implemented at DLR for Links and Nodes. The block is responsible for subscribing to a topic of LN and feeding received data into the Simulink controller

as well as publishing data to the LN topic of the other direction every timestep. The subscriber functionality also implements a blocking mechanism that blocks the whole controller until a new message of that topic is received in LN.
Synchronization in Gazebo is implemented in the C++ plugin responsible for the control of the robot. It utilizes a main loop to implement functionality called every timestep. When this method gets blocked in execution, it also blocks any further timesteps, while still rendering the 3D view.
The Gazebo subscriber mechanism is on a separate thread so Gazebo can run without blocking when the controller is not used. This also allows Gazebo to initiate the communication between Simulink and Gazebo by publishing the data required by the controller every timestep regardless of whether Simulink is running. This is necessary because Simulink's blocking mechanism does not allow for the sending of an initial message.
Multiple synchronization variables are introduced to implement the blocking mechanisms in C++:

- *int n_simulink_packets* - Counts up for every received valid Simulink packet, starting at zero.
- *condition_variable simulinkSync* - A C++ primitive that allows blocking until a condition is fulfilled
- *bool readyForStep* - Checked by the condition variable, prevents the main thread from executing before Simulink data is received
- *mutex dataReadyMutex* - Mutex for acquiring access to data

The blocking mechanism functions as follows in pseudocode:
Reader Timestep:

```
data = read()
lk = lock(dataReadyMutex)
if n_simulink_packets > 0:
    // atomically unlocks lk, waits until readyForStep is false and
    // acquires lk again
    simulinkSync.wait(lk, readyForStep == false)
copy(data, globalData)
n_simulink_packets++
readyForStep = true
unlock(lk)
simulinkSync.notify()
```

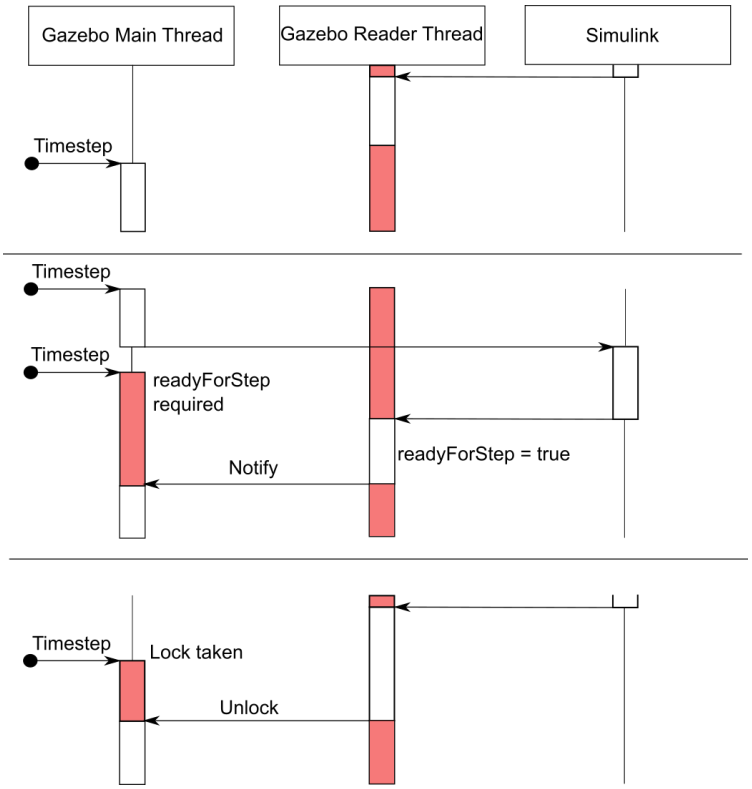


Figure 3.2: Blocking scenarios between Gazebo and Simulink, red areas mark blocking of execution

Gazebo Timestep:

```
lk = lock(dataReadyMutex)
if readyForStep == false:
    // atomically unlocks lk, waits until readyForStep is true and
    // acquires lk again
    simulinkSync.wait(lk, readyForStep == true)
SendToSimulink(globalData)
if n_simulink_packets > 0:
    readyForStep = false
unlock(lk)
simulinkSync.notify()
```

The mechanism for initiating communication is slightly hidden in both cases: As long as the reader thread doesn't find any messages, it stays blocked on the read() call.

Therefore *n_simulink_packets* remains 0, which prevents general Gazebo timestep from setting *readyForStep* to false.
For the reader thread, it's only useful to wait for *readyForStep* to become false if the main thread has a way to set it to false. In the first successful read(), $n\_simulink\_packets = 0$ so the main thread won't set it to false.
Since Simulink always waits for a packet before sending one, we can be certain that after a correct start, the nth Simulink packet will be sent strictly after the nth Gazebo packet (counting from the first packet that is answered). The only unknown and irregular variable is the start of a Gazebo timestep, which leads to three scenarios visible in Fig. 3.2 that are all covered by the described synchronization mechanisms:
Either the timestep starts after the reader thread finished copying the received data which obviously doesn't require any blocking, or the new timestep starts before the Simulink answer to the previous data arrived, in which case the thread will wait for *readyForStep* or the timestep starts while the reader thread has the lock on *dataReadyMutex* which makes the lock itself the blocking mechanism.

3.5 Controllers

The Simulink controller provides multiple different controller modes for different use cases relevant to the Rollin' Justin robot, these are all accessible in the digital twin as well. This allows any operation on the real robot to be simulated regardless of the mode required for it. Since Rollin' Justin is a humanoid robot designed to interact with humans and take over tasks for them, some of these modes contain mechanisms to react softly to collisions with external objects rather than reaching the desired position at any cost like industrial robots do.

- Zero Torque Control: No forces are applied to the joints apart from those necessary for avoiding self-collision and for resisting gravity. This mode holds the position of the robot and can't be used to move it to desired poses by commanding an interpolated movement. Instead, it is mainly used to easily move the joints by hand without a force pushing the robot back into a previous position, it simply follows the outside forces as long as they are not making parts of the robot collide with each other and stays in position once the outside force stops.

- Joint Impedance Control: The joints are treated as mass-spring-damper systems, resulting in a "soft" behavior when colliding with the environment or a person applying some force to the robot, e.g. shaking its hand. In this mode, the force against moving a joint angle by hand will grow the further it is from its desired position according to the stiffness and damping parameters. This mode

is particularly useful for calculated, careful contacts with the environment to prevent damage to the robot itself or humans interacting with it.

- Cartesian Impedance Control: This mode exploits a hierarchical approach to reach two goals at once: As the first priority, the commanded cartesian position of the hands and the upper torso in relation to the platform should be reached and held. Since the arms make up 7-DOF systems, there is one degree of freedom remaining when holding the hands in position which is referred to as the nullspace of the first part of the controller. As the secondary functionality the arm joints attempt to reach and hold their commanded angles while keeping the hands in position, this uses an impedance controller and prevents the last degree of freedom from drifting in the nullspace. The Cartesian Impedance Control mode is useful for scenarios with a focus on surface contacts, like cleaning a solar panel by applying a moderate amount of force.

- State Space Control: The current pose is interpolated towards the desired pose, on the real robot the result is sent to low-level joint controllers that control the torque individually for each joint. On the simulated robot, this is implemented with joint-level PD controllers directly in Gazebo. The State Space Controller is the only controller that requires Simulink's desired angle on the side of Gazebo instead of the calculated torque. This mode does not have a soft reaction to environment contacts, so forces acting on objects in the way of a movement might damage them or the robot, however this also makes the controller the most accurate. It is used for interacting with objects precisely, like picking up a cup or grabbing a handle.

For simulation purposes, the zero torque control mode seems of rather little use, since there is no human moving the joints there. The interesting usage of the simulation is for the robot to reach some larger objective, like picking up an object, by moving to given or calculated poses in one of the other three control modes. An overview over the respective Simulink and Gazebo behavior for each controller is given in table 3.3.

3.6 Digital Twin

The digital twin of Rollin' Justin is modelled in SDF (Simulation Description Format), a more general alternative to the common format URDF (Unified Robotics Description Format) since SDF is used by the Gazebo simulation system to describe mechanical constructions made of joints and links.
The pre-existing model was created in accordance to the measurements of Rollin' Justin's hardware elements by using the same composition of joints and links and

Table 3.3: Controller behavior

Controller	Simulink Output	Gazebo Actions	Observed Behavior
Zero Torque Control	Torques compensating gravity	Apply torques	Very slow drifting of arms
State Space Control	Desired position and torques compensating gravity	Set desired position of low-level joint controllers	Accurate replication of poses and movements
Joint Impedance Control	Torques compensating gravity and towards goal	Apply torques	Accurate replication of poses and movements
Cartesian Impedance Control	Torques compensating gravity and towards goal	Apply torques	Accurate replication of poses and movement

calculating kinematic and inertial properties from the digital 3D representations of the hardware elements.
We added static joint friction to all joints to prevent noise due to the discrete timesteps of the simulation. It prevents and finishes the movement of a joint once the change in position is small enough to consider it standing still and thus avoids joints oscillating closely around their goal position.
The following sections describe the further improvements made to make the digital twin suited for the usage with torques and explains the reasons behind them.

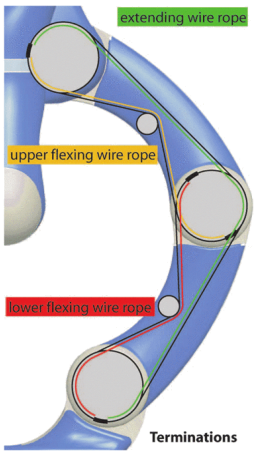
3.6.1 Reference Frames

As Rollin' Justin is nearly symmetrical, the arms of the digital twin are modelled in a mirrored way, with the left arm being a symmetrical opposite of the right arm mirrored along the vertical middle axis of Justin. However, Simulink uses a different reference frame regarding the angles of joints.
In the Gazebo reference system, when all joints are in their default position - 0 degrees - the second joint from the shoulder lifts the arm up and down such that 'up' is a positive angle and the motors require positive torques to hold the arm up and 'down' is the opposite.
For the left arm, this reference frame is inverted at the input and output of Simulink, because the real Justin requires this inverted reference frame due to its motor orientations. The controller internally calculates with the same reference frames as Gazebo

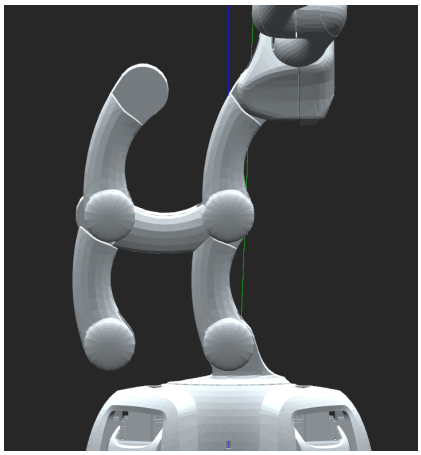
and only inverts the left arm values at the input and output interfaces. 'Up' is then considered a negative angle and requires negative torques for holding.
To keep the controller as is, the inversions need to be handled on the side of Gazebo: The measured angles of the left arm communicated to Simulink need to be inverted to replicate Justin's reference systems and the other way around the desired positions and torques commanded by Simulink need to be inverted in the Gazebo plugin before applying them to the joints.

3.6.2 Parallel Torso Construction

In order to isolate the torso from the arms such that their torques do not influence each other, the fourth torso joint is not actuated on the real robot, but coupled to the second and third torso joint by a tendon construction connecting it to the base of the robot. This construction mechanically compensates the orientation of the second and third torso joint to keep the uppermost fourth torso link parallel to the ground at all times. Any torque along the axis of the fourth torso joint is redirected to the base, which also gives the arm movements a stable base instead of transmitting some of the torques into the torso, this construction is explained in more detail in [24] and shown in Fig. 3.3(a). Since Gazebo only works with joints and does not have a concept of tendons, a



(a) Justin's physical tendon construction, image from [24]



(b) Link-Joint construction visualized in Gazebo

Figure 3.3: The two torso constructions fulfil the same purpose with different mechanical concepts

construction fulfilling the same purpose with only joints and links is required. It has to

keep the fourth torso parallel to the ground without blocking or impairing the joints below it.
The second, third and fourth torso joint combined have two degrees of freedom: Tilting the second or third joint has to lead to an equal tilt of the fourth joint in the opposite direction to keep the orientation of the fourth torso link.
This can be achieved with a parallelogram: Given a physical body that has some horizontal extension, it can be constrained to translation in a circle (one degree of freedom) by putting revolute joints on its horizontal ends and setting up parallel vertical links of the same length connected to the joints, these links also get connected to the ground with revolute joints of the same axis.
Rotating one of the ground joints forces the other one to follow at the same angle since the attached links hold an element at the top which is constant in length. This results in the joints attached to the horizontal object being at the same height at all times, fulfilling the requirement of holding it parallel to the ground, see Fig. 3.4.
This construction also transmits any torques applied to the horizontal object into the base the holding links are connected to.

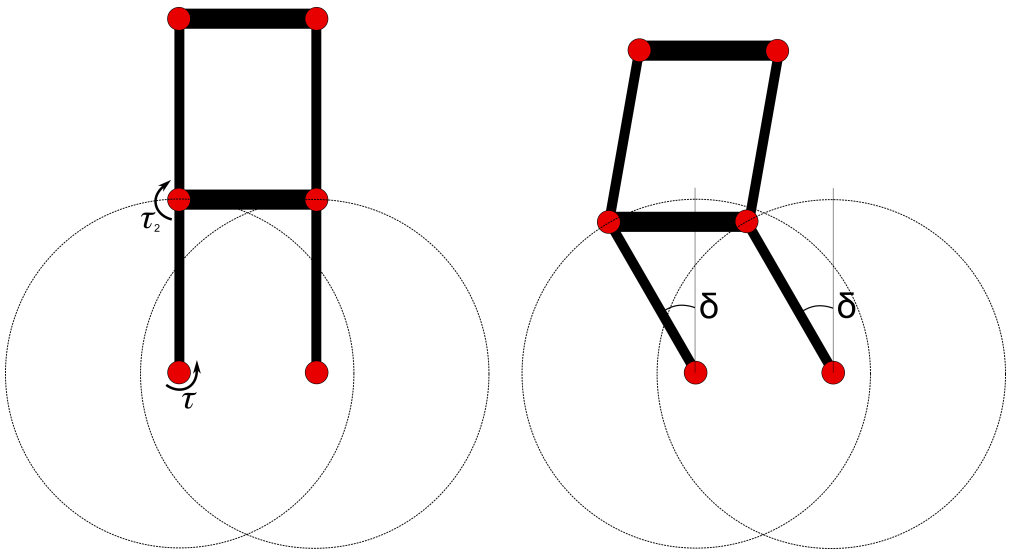


Figure 3.4: The conceptual construction of the parallel torso pieces. The actuated joints are marked with τ

Since both the second and third joint's orientation should be compensated, the construction is repeated downwards, where it is connected to the robot's base rotating around the vertical axis.

For the digital twin, the horizontal piece at the top is the fourth torso link, shown in Fig. 3.3(b).

3.6.3 Finger Controller

The control of Justin's fingers is separated from the controller of its body, this controller was not coupled to the simulation, but recreated as a PID controller inside Gazebo. This is due to the separation in reality, the Simulink controller for the hand is separate from the body controller and is a PID controller itself.
Interpolation of the finger positions is still done in the main controller and therefore the desired positions can be taken from the communication interfaces that were set up earlier. The fingers on the real robot do not have torque sensors but only actuators and position sensors, however the torques required to move the light finger links are very small.
Since the fingers have a very low weight, their behavior is not as consistent as that of the body joints in reality. As such, the main focus in their simulation design was on adjusting the mechanical parameters of the joints and links, such as their inertia tensors and static joint friction values.
The fourth finger pieces at the front representing the fingertips are connected to their predecessors through a passive hinge joint which follows the angle of the third finger joint through a tendon construction. This mechanism is not replicated in the simulation but replaced by another PD controller that actuates the fourth joint moving it towards the position of the third finger joint which suffices for the purposes of the simulation.
Gripping and lifting objects is one of the challenges that fully fledged physics engines such as ODE have difficulties with due to the large amount of contacts to be resolved and the inaccuracies in collider geometry. This was not a focus of this thesis, an approach to control gripping behavior is implementing a separate simulation that is focused on the detection of finger-object contacts to determine whether the forces justify viewing the object as gripped, as explained in [25].

4 System Evaluation

The accuracy of the behavior of the digital twin is of great interest. This does not refer to accuracy in the sense of following some desired path as strictly as possible or designing an ideal controller regarding some metric, but rather replicating the movement that the real robot shows under the same circumstances as closely as possible. The term error will in the following refer to the difference between the measurements of the real robot and those of the simulation.
We examine the pose and tracking accuracy in movements without contacts first and follow with an experiment with environment contacts. A short performance evaluation is at the end of the chapter.

4.1 Metrics

Multiple metrics need to be taken into consideration to gain a full picture of the simulation accuracy. The joint positions were held with complete accuracy before as they were directly set through kinematic methods. Now that they are a result of torques, both torque and position errors are analyzed as well as the end effector pose which is the result of the forward kinematic calculations through the robot's torso and arms to the hands.

4.1.1 Joint Positions

The metric chosen for joint position inaccuracies is the root mean square of arm and torso joint errors.
Let q_P be the vector of joint angles of the physical robot and q_S the vector of joint angles of the simulated Justin, with all angles given in degrees. The error vector e is calculated:

$$e_i = q_{P_i} - q_{S_i} \tag{4.1}$$

Let N be the number of compared joints, using the error vector we calculate the root

mean square error (RMSE):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (e_i)^2}{N}} \tag{4.2}$$

The RMSE does not have a unit even though in this case its input are angles in degrees, but it does have a property that can help with estimations regarding the significance of a high RMSE: If every joint has the same absolute error size y°, the RMSE will be y as well.
Compared to the mean absolute error (MAE), which is calculated by

$$MAE = \frac{\sum_{i=1}^{N} |e_i|}{N} \tag{4.3}$$

the root mean square penalizes outliers and variance in the data more which is desirable since outliers - i.e. joints that have significantly larger errors - would be subject to further investigation regarding their error cause [26].
A downside of this approach is that all joints are compressed into one metric, which leaves no possibility to deduct the distribution of errors over the different joints. For example, having a few joints with large errors and all other joints with almost no error can result in the same RMSE as every joint having a moderate error.
Thus, as an additional metric, the absolute joint errors are calculated separately and evaluated for a maximum error to grant some insight to the composition of the RMSE and a 'worst case' joint behavior.

4.1.2 Torques

Since a main advantage of the coupled controller is the integration of torques and forces into the simulation, their accuracy is of great interest as well. The joint position errors and torque errors are not necessarily proportional, it is even possible to have good pose accuracy in a trajectory while applying significantly different torques than the real robot, which would obviously lead to incorrect contact forces as well.
As the metric the RMSE (4.2) is used again, with the error defined as the difference between recorded and simulated *tau_act*, i.e. measured acting torques given in Nm. An absolute error of $1Nm$ on every joint results in an RMSE of 1.
A maximum absolute torque error among all joints is also calculated.

4.1.3 End Effector Poses

Another approach to determine the accuracy of the robot's behavior is a comparison of the poses of its end effectors - i.e. the robot hands - as these are in the focus of

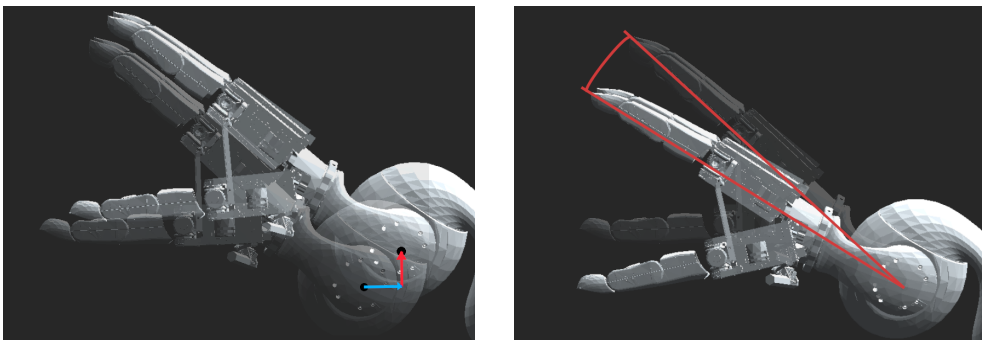


Figure 4.1: End effector cartesian position and rotation error visualized

many practical uses of Justin. More precisely, the position and rotation of its wrists are analyzed with this metric, not its finger poses, visualized in Fig. 4.1. This measurement is especially interesting in the case of the cartesian impedance controller, which utilizes inverse kinematics to guarantee fine control over the end effector position, rotation and therefore contact force.
Such a pose consists of two different components, its cartesian position in space and its rotation, both given in the reference system of the robot base. These can not be compared usefully in one metric because of their different units and are therefore measured separately.
Since the pose of an end effector is given as a transformation matrix by the real robot in the reference system of its base platform, we first filter out a position vector with three elements - corresponding to X, Y, Z in the base reference system - and a unit quaternion representing the rotation with four elements.
The Gazebo simulation environment offers an easier way to obtain this data by exposing the world pose of objects - i.e. their position and rotation - in a unified structure. Calling $WorldPose()$ on a link returns a $Pose3d$ that lets one get its world position through a call to $Pos()$ while $Rot()$ returns the rotation as a unit quaternion.
Since the robot's logged end effector pose is in the base platform's reference system, the simulation data also needs to be transformed in Gazebo to represent the position and rotation in the reference system of the base.
Gazebo offers this with subtraction of poses, i.e. $hand.WorldPose() - base.WorldPose()$ returns the desired pose.

The position accuracy metric Δv is calculated using the euclidean distance of simulated and real cartesian position for both end effectors:
with $i \in \{left, right\}$:

$$v_{sim\_i} = \begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix} ; v_{real\_i} = \begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix}$$

$$\Delta v_i = |v_{sim\_i} - v_{real\_i}| = \sqrt{(x_s - x_r)^2 + (y_s - y_r)^2 + (z_s - z_r)^2} \tag{4.4}$$

The average of left and right distance is chosen as the metric, but since both hands behave the same and are constructed symmetrically, their cartesian errors are very similar.
This pose data can be taken directly from Gazebo as described above to get the mathematically accurate pose in the simulation using the Gazebo robot model. This end effector position will be referred to as the 'Gazebo position'.
Another source of the end effector pose data in the base reference frame is the logged data of the Simulink controller running with the simulation.
It utilizes the same forward kinematics calculations as the real robot and uses its own robot model internally. Since this model is used identically on the real robot controller, the cartesian position error is reduced to the accumulated offsets caused by joint position errors. This end effector position will be referred to as the 'Forward kinematics position'.
Both of these approaches will be analyzed in the results with a more in-depth explanation in section 5.1.3.

We chose to only evaluate the end effector rotations from Gazebo as the two approaches give practically identical results for rotations.
A lot of different methods exist to obtain a metric for rotational distance in three dimensions that are often functionally equivalent. The chosen method is the metric 5 from [27], which is dimensionless, bi-invariant and a metric on the Special Orthogonal Group $SO(3)$ that 3D rotations form. It can be simplified for usage with unit quaternions to use just 7 multiplications and one square root making it quite efficient and easy to implement.
The rotational distance θ is calculated from the two unit quaternions $q1$ and $q2$ with:

$$\theta = 2\sqrt{2(1 - (q1 \cdot q2)^2)} \tag{4.5}$$

with the $\cdot$ being a dot product of the quaternion vectors with four elements.

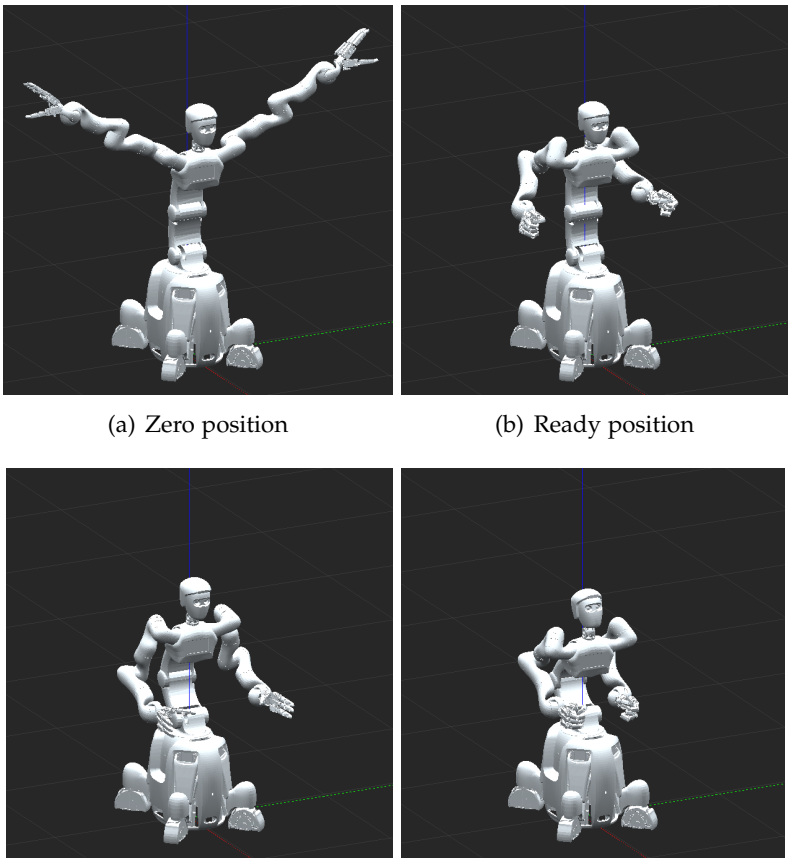
4.1.4 Finger Positions

The hand controller is implemented only in Gazebo, meaning it is almost independent of the Simulink model apart from the interpolation of desired joint positions. The finger torques are not measured with sensors on the real robot and are dismissed for the measurements, the finger positions are evaluated just like other joint positions.
Since the movements and poses are symmetrical for both hands the 24 actuated finger joints are combined into the RMSE metric (4.2), the fourth finger joints are not actuated on the real robot but follow the angle of the third ones through a tendon construction. They are therefore ignored for these experiments.

4.2 Experiment Setup

All measurements are reproducible independently of hardware since timesteps are calculated with a fixed length. On the software side, Gazebo 11.0.0 is used with the ODE physics engine. The Simulink controller of Justin is compiled in Matlab 2018b for OSL15.
Where not explicitly stated otherwise, measurements were conducted with the state space and joint impedance controller.
We recorded poses and movements on the real robot and logged its telemetry data with 1000 samples per second. This data contains all the required fields such as positions, torques and end effector poses.
In the simulation, the experiments were conducted under the same conditions, logging similar telemetry data at 1000 samples per simulation second which can deviate from real time, this relation is also analyzed in section 4.5.
For the following accuracy measurements we utilized a set of common poses of the humanoid robot:

- A zero position where all arm joints have a position of zero degrees and are forming straight lines upwards in a V-form (Fig. 4.2(a))
- A ready position with a straight torso and outstretched hands (Fig. 4.2(b))
- An idle position with the hands further back and the torso slightly bent (Fig. 4.2(c))
- A parking position with the torso bent far and the hands even further back, minimizing the size and height of the robot (Fig. 4.2(d))



(a) Zero position (b) Ready position

(c) Idle position (d) Park position

Figure 4.2: The main robot poses used for the experiments

4.2.1 Parameters

The state space controller is implemented in Gazebo as a PD controller and receives the proportional gain directly from Simulink. To reduce noise and increase accuracy, the derivative term was empirically evaluated to a value of 3 for the arm joints while the torso joints work best with a derivative gain of 50. These values were used across all following measurements.
Since the head of Justin is not at the focus of attention for most tasks and is always position-controlled, it was also implemented as a PD controller, a reliable combination of parameters was identified and used in the experiments. However the head's accuracy was not evaluated and it does not influence the behavior of other parts of the robot in a significant way. P gain was set to 50, D gain was set to 1.

The finger joints used a PD controller with the same parameters for each finger with the proportional term at 3.5 and the derivative term at 0.002.

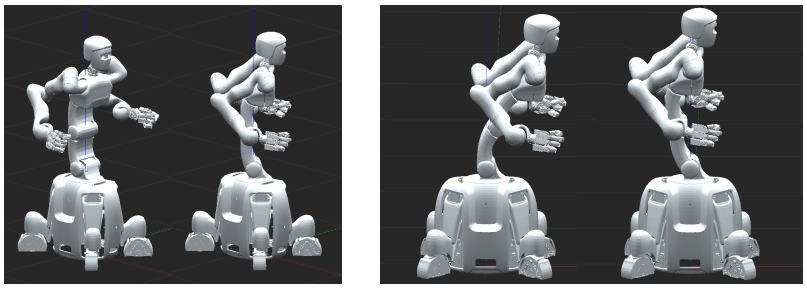
4.2.2 Static Pose Accuracy

Static pose accuracy refers to the errors between Justin's positions and torques and the simulated ones while holding one of the poses since inaccuracies in e.g. inertial properties of the model can lead to discrepancies. The torques and angles are practically noise-free for holding static positions, the error metrics are still averaged over five hundred samples to avoid any outliers.

4.2.3 Tracking Accuracy

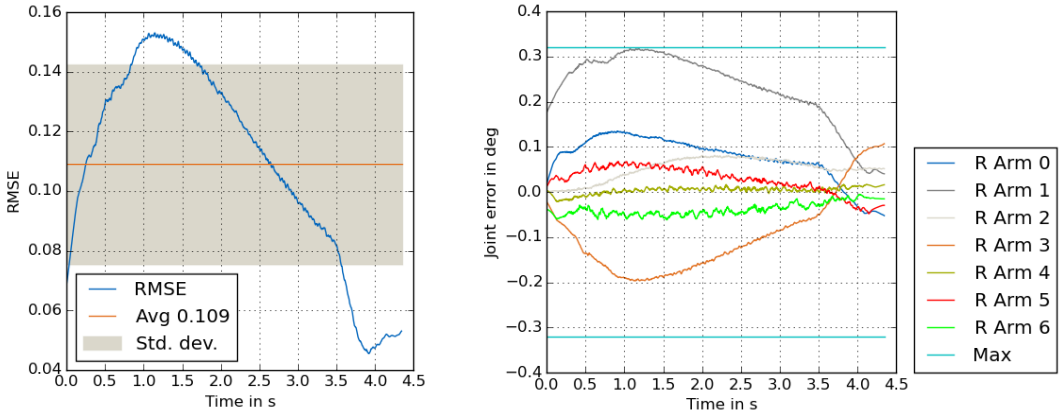
For the tracking accuracy we defined multiple movements to draw meaningful conclusions.
Six movements are defined between the neighboring poses of Fig. 4.2 in both directions, e.g. from 'zero' to 'ready', 'ready' to 'idle' etc.
These provide large motions of the arms, for the torso accuracy evaluation two additional movements were used:
The first and lowest torso joint rotates the entire robot body around its vertical axis and isn't required in any of the previous trajectories. The movement *rotate_torso*1 rotates the first torso joint from 0° to 45° while the rest of the robot holds the 'ready' position. Start and end pose are visible in Fig. 4.3(a) from the same perspective.
Because of the large torques required to hold the upper body against the gravity, it is of great interest to analyze larger movements of the corresponding torso joints 2 and 3. We define the movement *torso_back* to tilt the second torso joint backwards from 15° to -15°, seen in Fig. 4.3(b).
These two movements were not used for the measurements of arm joint accuracies.

The comparison of data points over time results in error graphs, which can be useful for more detailed insight into the temporal distribution of the errors.
For easier comparability we calculate a mean, standard deviation and maximum of the error metric over time for each movement. This reduces the temporal data to its most representative attributes and serves as the basis for comparing controller accuracy. For the metrics utilizing RMSE we additionally determine a maximum error of the single joints over the whole time of the movement. An example for the extraction of a mean RMSE and a maximum single joint error is seen in Fig. 4.4.
While this removes the temporal dimension of the data, this information reduction still allows us to gain relevant insights into tracking accuracy. The maximum error can



(a) *rotate_torso*1 movement (b) *torso_back* movement

Figure 4.3: The special torso movements



(a) RMSE of arm joint positions with mean and standard deviation

(b) Separate joint errors of the right arm in degrees with markers for the maximum error

Figure 4.4: Arm Joint Error Metrics over time for the *zero_to_ready* movement in state space control

guarantee a minimum degree of accuracy throughout the whole movement.

A challenge that arises when comparing logged and simulated data over the timespan of a movement is synchronization since the data logs are started manually some time before the movement begins. Including these parts may skew results since the errors of those poses may differ from the errors along the trajectory, the pose errors are already evaluated in the pose accuracy measurements.
Synchronization is achieved through the commanded joint angles in the logged data. These are identical for the physical and simulated movement which provides us with a

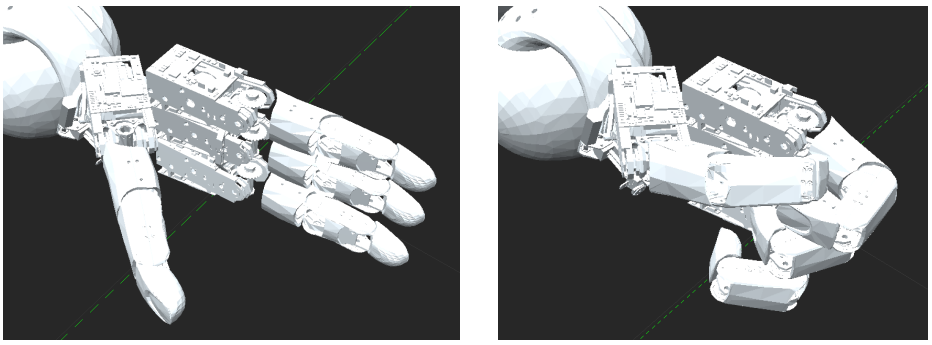


Figure 4.5: Open and closed finger poses used for the experiments

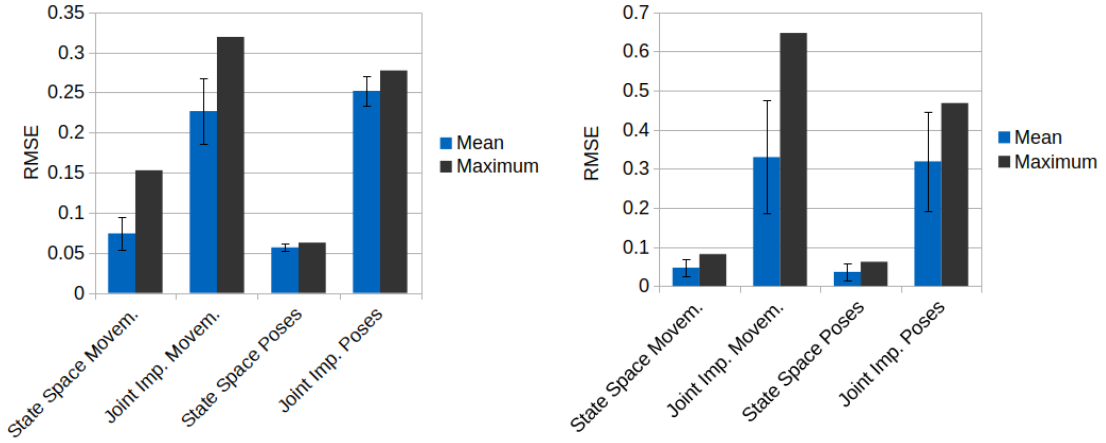
shared movement phase of the same length. For the detection of movement start and end points, a minimum deviation of *q_cmd* from the joint angle commanded at the start and end of recording is demanded respectively. To account for this threshold we keep two hundred samples before the threshold is reached and since the joints follow the commanded angles with a varying delay, four hundred samples (i.e. less than half a second of data) are kept after the calculated end.

4.2.4 Finger Accuracy

The fingers are not evaluated with the four robot poses but with their two poses of interest, which is an open hand and a closed hand to enable gripping and dropping of objects, see Fig. 4.5. The experiments are conducted for these two static poses and tracking measurements are taken between them.

4.3 Experiment Results

The results of the experiments are presented for the different metrics with a focus on the different controllers and their respective accuracy results. Depending on the metric, the difference between static and dynamic behavior is also in the focus.
The results for each movement were already compressed over the time domain into mean, standard deviation and maximum of the error metrics, the focus on controller modes and differences between poses and movements is achieved by compressing these results again:
For any controller there are six movements with a mean, standard deviation and maximum error term. The mean of the means is calculated, weighing every movement's mean error the same, no matter how long its source is in the time domain. The maximum mean refers to the maximum of those six mean values while the maximum error



(a) RMSE of arm joint positions over all movements and poses

(b) RMSE of torso joint positions over all movements and poses

Figure 4.6: Mean and maximum RMSE of arm and torso joint positions in both analyzed controllers

(or maximum RMSE) refers to the largest of the six maximum error terms, i.e. the highest peak of the error term at any point in time in any movement.
Additionally the diagrams presented in the following section show standard deviation markers which refer to the standard deviation among the six movements (or four poses) without respect to the time domain of the error metric, see Fig. 4.6.
Regarding joint position and torque accuracy, the measurements were divided to accommodate to the different underlying constructions: The arm joints are evaluated separately from the torso joints to gain insight into the accuracy of the parallel construction used for the torso described in section 3.6.2.
Additionally, most movements did not involve any changes of the first torso joint that rotates the whole body around its vertical axis. In these cases averaging the torso accuracy is slightly misleading as the first torso joint stays at a very low error. More attention should be paid to the maximum torso error that would indicate large errors of the two upper torso joints.

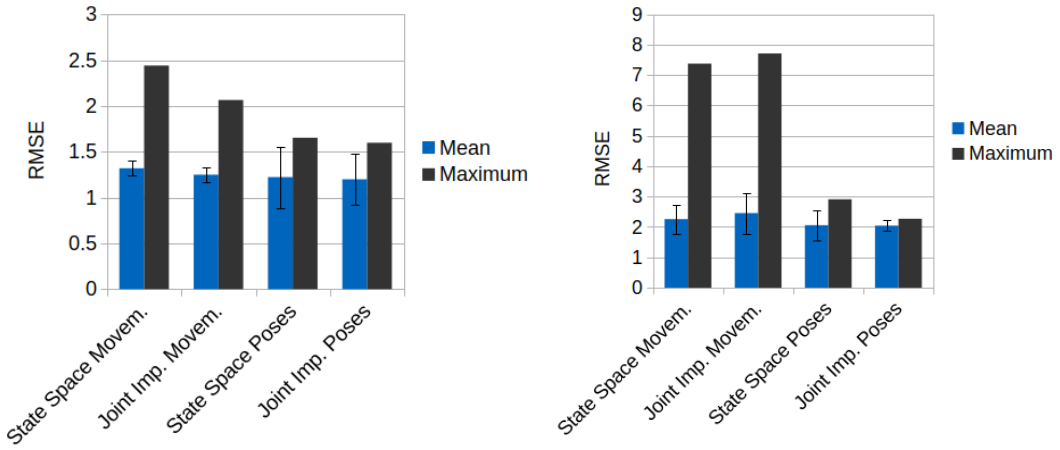
subsectionJoint Position Results We first take a look at the state space controller results and the joint impedance controller results afterwards as their position accuracy demands and results differ significantly.
The results presented in this section are listed in table 4.1, the mean and max RMSE values are visualized in Fig. 4.6. The diagrams also show the standard deviation among the mean RMSE values of poses and movements respectively.

Table 4.1: Mean RMSE of joint positions for arms and torso

	State Space Poses	Joint Imp. Poses	State Space Movement	Joint Imp. Movement
Arm Mean RMSE	0.0568	0.252	0.0742	0.227
Arm Max RMSE	0.0627	0.277	0.153	0.319
Arm Max Error	0.162°	0.673°	0.321°	0.797°
Torso Mean RMSE	0.0362	0.319	0.0469	0.330
Torso Max RMSE	0.0622	0.468	0.155	0.836
Torso Max Error	0.107°	0.744°	0.267°	1.413°

Starting with the poses, the arm joint errors result in a mean RMSE of 0.0568, no pose has an RMSE larger than 0.0627 and the maximum arm joint error is at 0.162°.
The mean RMSE of the torso joints is at 0.0362 for poses, the maximum RMSE of 0.0622 is reached in the parking position and any error of a single torso joint remained below 0.107°.
In the movements, the arm joints reach a higher mean RMSE of 0.0742 and a higher maximum RMSE of 0.153. No arm joint reaches an error larger than 0.321°. Both maximum values occur in the *zero_to_ready* movement, these measurements are also visualized in Fig. 4.4.
The torso joints have a mean RMSE of 0.0469 for the movements, their maximum RMSE lies at 0.155 and the maximum error reaches 0.267° in the *rotate_torso*1 movement.

For the joint impedance controller, we again start with results for the poses:
The arm joint errors result in a mean RMSE of 0.252 and a maximum RMSE of 0.277. One of the arm joints reaches a position error of 0.673°.
The torso joints are held with similar accuracy: A mean RMSE of 0.319, but a larger maximum RMSE of 0.468 in the parking position and a maximum joint error of 0.744°.
In movements, the joint impedance controller has a lower mean RMSE for the arms at 0.227 than for the four poses. The arm joints' maximum RMSE lies at 0.319, the largest position error observed on a joint is at 0.797°.



(a) RMSE of arm joint torques over all movements and poses

(b) RMSE of torso joint torques over all movements and poses

Figure 4.7: Mean and maximum RMSE of arm and torso joint torques in both analyzed controllers

The torso joints have a mean RMSE of 0.330 for movements in joint impedance control, however their maximum RMSE reaches 0.836 and the largest position error is observed at 1.413° in the *rotate_torso*1 movement.
A closer analysis of the joint positions over time shows that the position noise is negligibly small for both controllers, an example is seen in Fig. 4.4(b).

4.3.1 Torque Results

The results regarding torques are divided into pose and movement evaluation rather than controller modes since the results of the two controllers are similar in large parts. All results of this section are listed in table 4.2 and the mean and maximum RMSE values are visualized in Fig. 4.7.

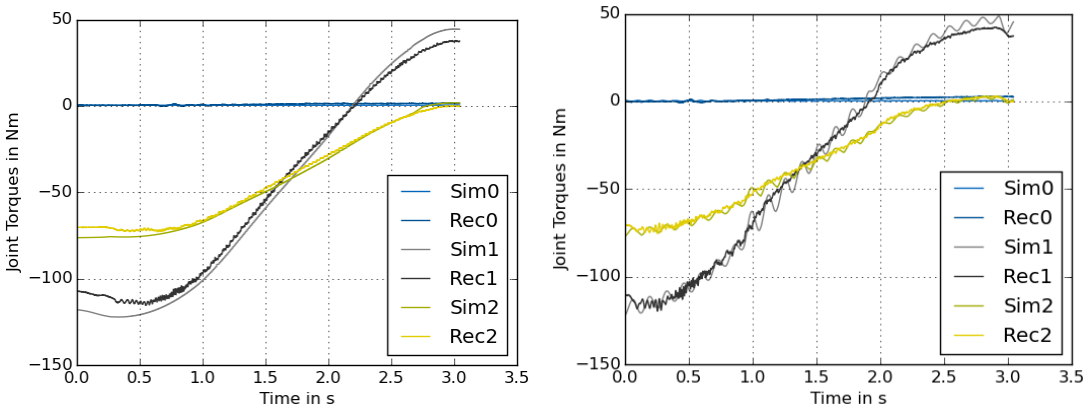
Starting with the poses, the arms show a mean RMSE of 1.223 in state space control and 1.199 in joint impedance control with maximum RMSE values of 1.653 and 1.598 respectively. The maximum torque errors are also similar at 3.293 Nm in state space and 3.210 Nm in joint impedance control.
The torques of the torso joints have larger mean RMSE for both control modes, in state space control the mean is at 2.057, in joint impedance at 2.044. The maximum RMSE is at 2.911 for state space control and 2.267 for joint impedance control. The state

Table 4.2: Mean RMSE of joint torques for arms and torso

	State Space Poses	Joint Imp. Poses	State Space Movement	Joint Imp. Movement
Arm Mean RMSE	1.223	1.199	1.320	1.250
Arm Max RMSE	1.653	1.598	2.441	2.065
Arm Max Error	3.293 Nm	3.210 Nm	6.245 Nm	5.091 Nm
Torso Mean RMSE	2.057	2.044	2.257	2.454
Torso Max RMSE	2.911	2.267	7.375	7.710
Torso Max Error	3.928 Nm	3.009 Nm	11.644 Nm	11.977 Nm

space controller also reaches a higher maximum single joint torque error at 3.928 Nm compared to 3.009 Nm for the joint impedance controller.

In movements, the arms show similar mean RMSE values as in the poses at 1.320 and 1.250 for state space and joint impedance control respectively, but their maximum RMSE values are significantly higher. State space reaches an RMSE of 2.441 and joint impedance control reaches 2.065 at some point over the course of the movements. The largest torque errors observed are 6.245 Nm in state space control and 5.091 Nm in joint impedance control.
The mean RMSE of the torso joints is slightly larger than for the poses at 2.257 in state space control and 2.454 in joint impedance control, the maximum RMSE is significantly larger at 7.375 and 7.710 respectively. The largest torque errors are at 11.644 Nm and 11.977 Nm, both of these occur in the *torso_back* movement, the course over time is visible for both controllers in Fig. 4.8.
Significant oscillations in the torques can be observed when moving torso joints in state space control, which are stronger in simulation than reality, see Fig. 4.8(b).



(a) Torso torques in joint impedance control

(b) Torso torques in state space control

Figure 4.8: Torso torques per joint in the *torso_back* movement in both controllers, recorded and simulated data in similar colors

4.3.2 End Effector Pose Results

Two different ways of determining the end effector positions in the simulation are evaluated in the following. The Gazebo measurements and Simulink forward kinematics both offer access to the position relative to the base platform, but use different models for the calculation.
The focus of these results lies on the end effector positions calculated by the forward kinematics inside Simulink as they are calculated the same way for the real robot. Simulated and real poses are therefore free from possible model discrepancies and use the same representation of the robot, simply accumulating the errors caused by different joint positions.

The state space controller produces significantly better results than the joint impedance controller with smaller mean and maximum cartesian errors: Poses in state space control result in a mean error of $0.171cm$ while joint impedance control reached $0.568cm$ mean error. Fig. 4.9(a) shows the mean cartesian errors in meters, where the state space controller also has smaller standard deviations compared to the joint impedance controller.
The mean cartesian error does not grow significantly in movements: The mean error stays at $0.171cm$ for state space control and rises slightly to $0.601cm$ for joint impedance control. However, the maximum position error of the end effectors at any point in time occurs in the *zero_to_ready* movement in joint impedance control with a cartesian error

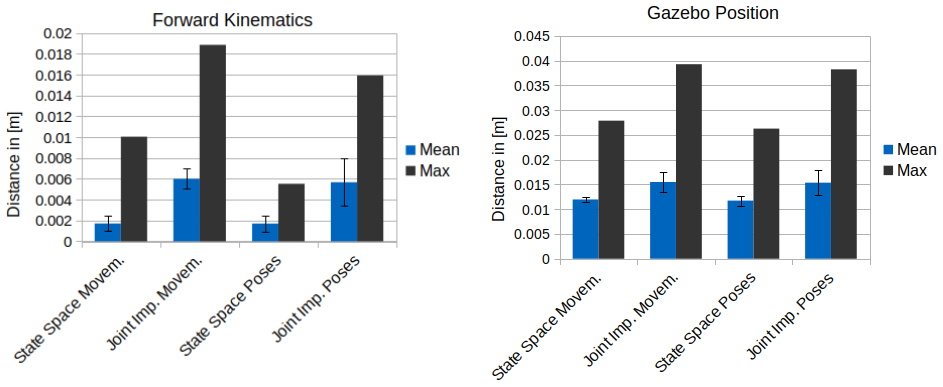
of 1.89cm. The maximum position error observed in state space control is at 1.01cm and occurs in the same movement.
The cartesian errors determined through the Gazebo position measurements are significantly higher, all mean errors are larger than 1cm. A detailed comparison is seen in Fig. 4.9.
The rotation errors of the end effectors are also crucial to correct interactions with the environment. While the metric is dimensionless itself, it is useful for the estimation of the occurring errors to compare them to a rotation about one degree on one axis: It leads to a rotation error of 0.0247.
The measured rotation errors show a significant difference between state space and joint impedance control, with the state space poses being held especially well with a mean rotation error of 0.00597 while poses in joint impedance have a mean error of 0.0152. The maximum rotation error at any point of simulation occurs in the zero pose with joint impedance control at 0.0229. It's also clearly visible from Fig. 4.9(c) that the controllers don't produce significantly larger rotation errors in the hands when movements of the robot are executed.

4.3.3 Finger Position Results

The finger joints reach an RMSE of 2.75 for the movement from open to closed fingers with a large standard deviation of 0.563 over the time of the movement, while the movement in the opposite direction only reaches 2.029 mean RMSE and a standard deviation of 0.254. The RMSE results for the two poses were 2.357 for the open hand and 2.0355 for the closed hand, see Fig. 4.10. The maximum error of any single joint occurs in the movement from open to closed pose with 6.925°, see Fig. 4.11.

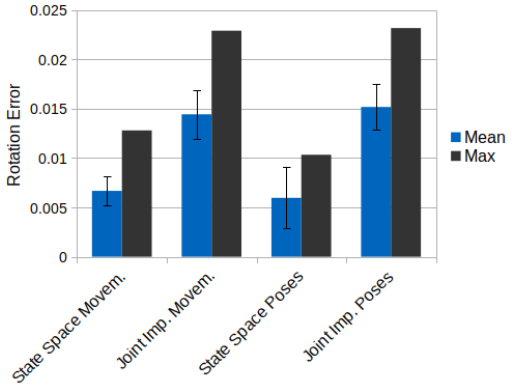
4.4 Environment Contact Experiment

Practical usages of the coupled systems include the prediction of contact forces with the environment, this was previously not possible when the robot wasn't controlled by torques. To get an estimation of the behavior of the robot when facing significant external forces, an experiment is set up in reality and simulation where the robot executes an action template. These templates define a sequence of atomic operations like short movements, setting a control mode or changing parameters. The sequence as a whole constitutes a robot action that represents some human task, like picking up an object [28].



(a) Cartesian error of end effectors from forward kinematics

(b) Cartesian error of end effectors measured in Gazebo



(c) End effector rotation error

Figure 4.9: Metrics for the end effector poses in state space and joint impedance control

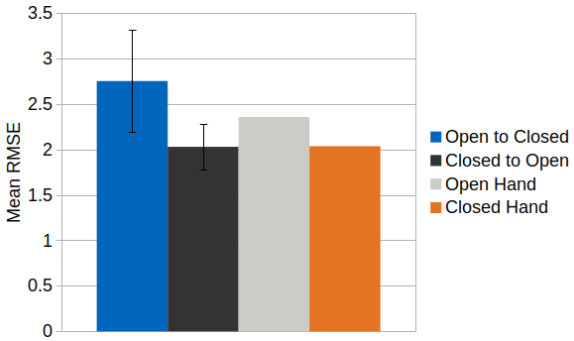


Figure 4.10: Finger Mean RMSE values for the open and closed finger pose and movements between the two

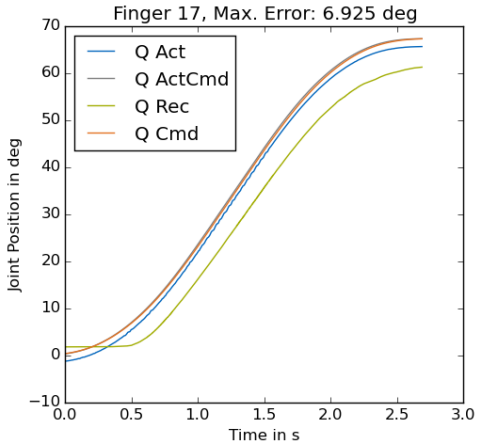


Figure 4.11: Finger 17 with the maximum position error in the hand closing movement

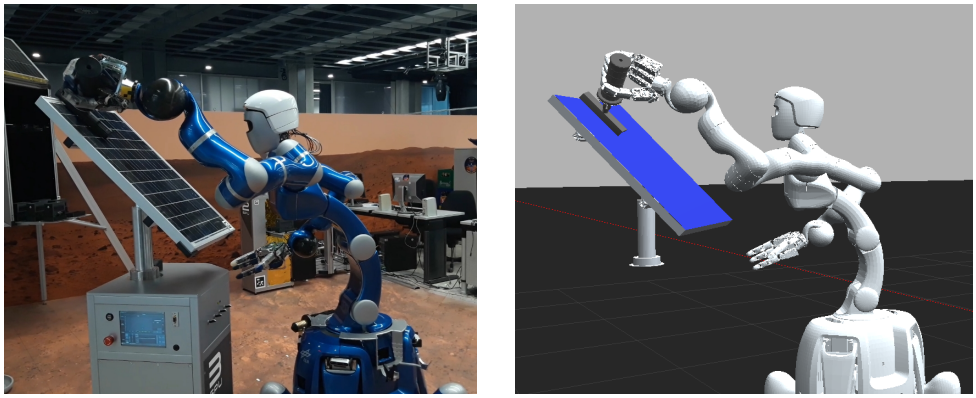


Figure 4.12: The real and simulated Justin wiping the solar panel surface

4.4.1 Setup

The Rollin' Justin lab is equipped with an environment that allows for different robot interactions and tasks that would be useful for extraterrestrial robotic services like connecting to a charging station, opening doors or adjusting the orientations of solar panels. In order to test the compliant contacts, we chose a wiping application in which Rollin' Justin utilizes a wiper tool to clean the surface of a solar panel from dust with three motions in full contact with the surface, see Fig. 4.12. This scenario is also commonly executed remotely by astronauts in their trainings with Justin.
The world state is analyzed by the real Justin through image processing and grants access to estimated positions and orientations of the solar panel and the robot. These

are replicated in the simulation.
The wiper tool is gripped by Rollin' Justin with all four fingers. This poses a challenge to the simulation since multi-contact resolving often doesn't suffice to hold an object inside the hand. Instead, the wiper receives a fixed connection to the hand link with an offset to approximate the position of the real wiper in Justin's hand.
This offset is a product of uncertainty since the relative pose of the wiper is not known and not entirely consistent in the real hand either. We approximated its pose manually and run the experiment with varying values for one of the rotation angles to determine an ideal approximation, in steps of 0.0, -0.075 and -0.15 in radians.
The position and orientation of the solar panel also introduce some inaccuracy. From our knowledge about the real scenario we can determine that the robot and the panel are only rotated about the vertical axis - even though small rotations about the other two axes are suggested by the image processing.
We also vary the vertical position of the panel between 1.00, 1.02 and 1.04 meters in the experiments. This way the desired end effector position is differently far behind the panel surface, resulting in different contact forces.

The experiment is conducted in the cartesian impedance controller (with the torso also in cartesian impedance control) because the compliant behavior should be applied in cartesian space - as an offset from the desired 3D position of the end effector - and not in joint space.
The concept of impedance control is suited very well for applications of this type: The actuation of the robot follows an underlying spring-mass-damper model that reacts to environment contacts impeding its position and velocity in a controlled way in contrast to position controllers. With the cartesian impedance controller, the contact force with a surface grows with the distance between the current and desired end effector position, i.e. the desired position of the wiper head should be behind the panel surface. The distance between the panel surface and the desired position can therefore be used as a parameter for the contact force.
The wiping application requires moderate contact forces, enough to wipe away dust without compressing the soft wiper head too harshly.
To replicate the desired movement in the simulation, the planned trajectories are reused as the commanded input to the controller, not taking the real occurring movement into account that is impaired by the environment contacts.

The following metrics are used to analyze the accuracy of the simulated scenario:

- A qualitative and quantitative approach to the external torques calculated by the controller as the difference between measured and commanded torques - i.e.

analyzing the shape and relative magnitude of torque peaks and comparing the absolute torques occurring.

- The left hand's cartesian position is analyzed to determine the accuracy of the simulated movements in a contact scenario. This metric uses the forward kinematics end effector position, not the one measured in Gazebo.

4.4.2 Results

The recorded movement of the real robot includes three contacts with the panel surface at different heights.
The different combinations of parameters do not all result in contacts between the wiper and the panel surface, but for each evaluated x-angle a sufficient panel height leads to three contacts. The parameter combinations and their respective contacts in the simulation are listed in table 4.3.

Table 4.3: Simulated solar panel contacts for different parameters sets

	1.00m	1.02m	1.04m
Wiper x-Angle 0.0	2/3 contacts	3/3 contacts	3/3 contacts
Wiper x-Angle -0.075	no contacts	3/3 contacts	3/3 contacts
Wiper x-Angle -0.15	no contacts	no contacts	3/3 contacts

The time discrepancies in the commanded joint angles were consistent among the different parameter sets with one larger offset of four seconds - i.e. the simulation continued a movement immediately while the real robot stayed at a position for four seconds. For some steps, no offset was detected, some further minor offsets of 0.05s to 0.5s were found. In all cases, the simulation was ahead of the real robot's commanded angles.
These time discrepancies were compensated in the simulation data by artificially delaying following data points by the measured offset. This was achieved by logging the execution timestamps of the consecutive action steps in the simulation and finding the same commanded joint positions in the recorded robot data at later points in time.

Looking at the external torques influencing the left arm joints qualitatively, there

are some deviations outside of contact phases in the calculated external torques of $\pm 2Nm$.
The three torque peaks recorded on the real robot affect the first and fourth arm joint the most.
The first joint experiences external torques of 14Nm for the first contact, 12Nm for the second and 8Nm for the third one. The fourth joint experiences torques of 8Nm for the first two contacts and 6Nm for the third. The third contact consistently results in smaller torques across all joints.
In the simulations with parameter sets leading to contacts the torques occurred at the same points in time relative to the action start and showed steeper curves with narrow peaks while the recorded external torques roughly remained at their maximum for about one second.
The qualitative analysis of the graphs also shows that for every simulated experiment, the torques produced by the panel contact affect the fourth joint much more than the first, the opposite is the case for the recorded torques. An example is visible in Fig. 4.13, where the simulated torque peaks are similar to the recorded ones for the first arm joint and four times larger for the fourth one.
The main focus of the external torques lies on the peaks, a quantified evaluation for each parameter set can be found in table 4.4.
It's obvious from the results that for each wiper angle the lowest panel height necessary to produce three contacts creates more accurate torques than larger panel heights. The torques in the simulation were roughly at half the magnitude on the first joint and 50% larger on the fourth joint in all three of these parameter combinations.
In cases like the one displayed in Fig. 4.13 where the panel is even higher, the larger torques are mainly directed to the fourth joint and produce very large peaks. In the example figure, almost -40Nm are reached at a joint that has a torque limit of 100Nm. Examining other joint torques for that parameter set shows the same relationship of 2-4 times larger torques, which strongly suggests that the contact force in the simulation is also significantly larger than in reality.
A direct measurement of the contact force is not possible in the real solar panel experiment.

The desired cartesian end effector positions are fully synchronized except for the time offset compensation areas.
It is obvious that the desired end effector position of the left hand can not be reached with the panel in the way.
A closer look at the measured cartesian positions shows that there are small time offsets during the movements where commanded positions are synchronized, an example is shown in Fig. 4.14. These discrepancies produce peaks in the cartesian errors,

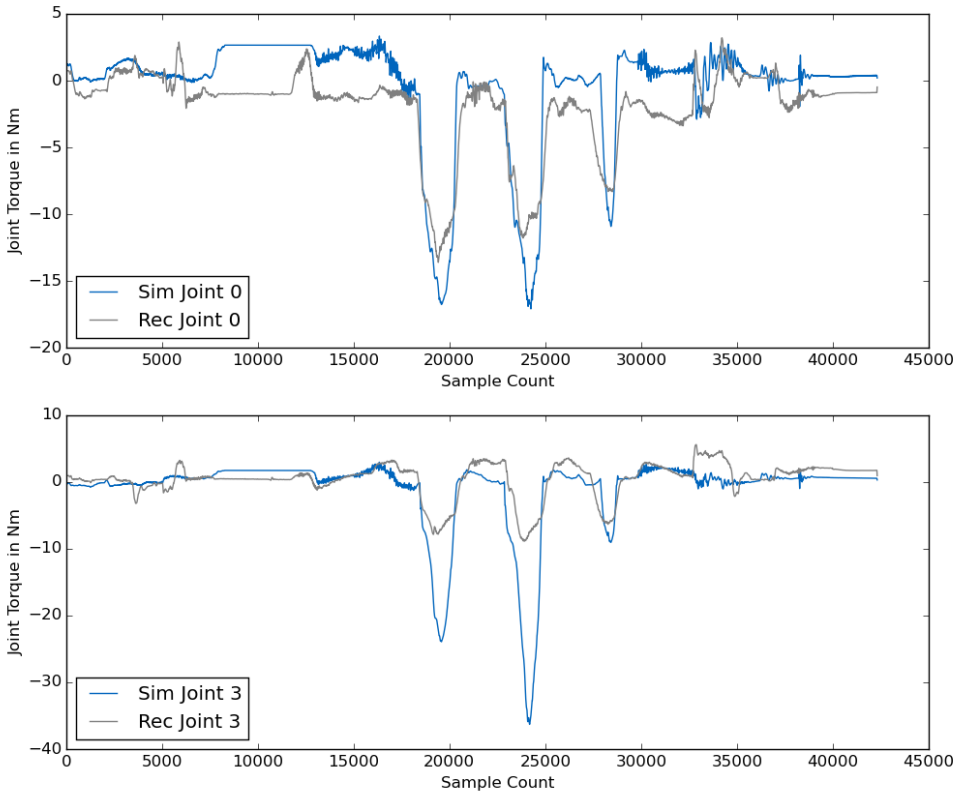


Figure 4.13: Recorded and simulated torques from the panel contact on the zero-indexed joints of the left arm, panel height 1.04m, wiper angle -0.075 rad

however the positions do reach very similar values at the end of their differently timed movements.
Even with the peaks caused by the time offsets, the average cartesian error over the contact phase of the task is less than 3cm for all parameter sets.

The z-position of the end effector is limited by the panel and stays nearly constant during the contact. Comparing this simulated z-position to the recorded one shows different discrepancies for the parameter sets. The commanded z-position is approximately 1cm lower than the one measured on the real robot. This is due to the panel blocking further movement, the depth beyond the panel surface is necessary for the

Table 4.4: Simulated contact torques for different parameters sets in relation to peaks recorded on the real robot, joints are zero-indexed

	Panel z-Pos 1.00m	Panel z-Pos 1.02m	Panel z-Pos 1.04m
Wiper x-Angle 0.0	Joint 0: 50% smaller torques, Joint 3: 50% larger on the second contact	Joint 0: 50% larger peak, but similar curve forms, Joint 3: 4 times as large	Joint 0: > 2 times as large, Joint 3: 5 times as large
Wiper x-Angle -0.075	no contacts	Joint 0: 50% smaller torques, Joint 3: 50% larger on the second contact	Joint 0: 2-4Nm larger peaks, Joint 3: up to 4 times as large
Wiper x-Angle -0.15	no contacts	no contacts	Joint 0: 50% smaller torques, Joint 3: 50% larger on the second contact

controller to create a force.
The parameter sets that lead to overly large torques in the joints also result in 1cm distance to the commanded z-position, while those with more realistic torques reach the commanded z-position quite exactly. This is especially clear when varying the panel height for the x-angle of 0, the highest panel position of 1.04m leads to the most accurate z-position, but the torques occurring in the arm joints are far too large, see Fig. 4.15.

4.5 Performance

Achieving accurate results from the coupled systems does not depend on the performance of the systems since they internally simulate constant time step lengths. It's still desirable when it comes to its usages: Not only should results generally be available as soon as possible, in live-action scenarios the simulation shouldn't run behind the real events but rather predict future movements and events. In the case of probabilistic predictions, multiple scenarios need to be run with small changes in the assumptions about the real world state. These simulations are only useful if their real-time relation

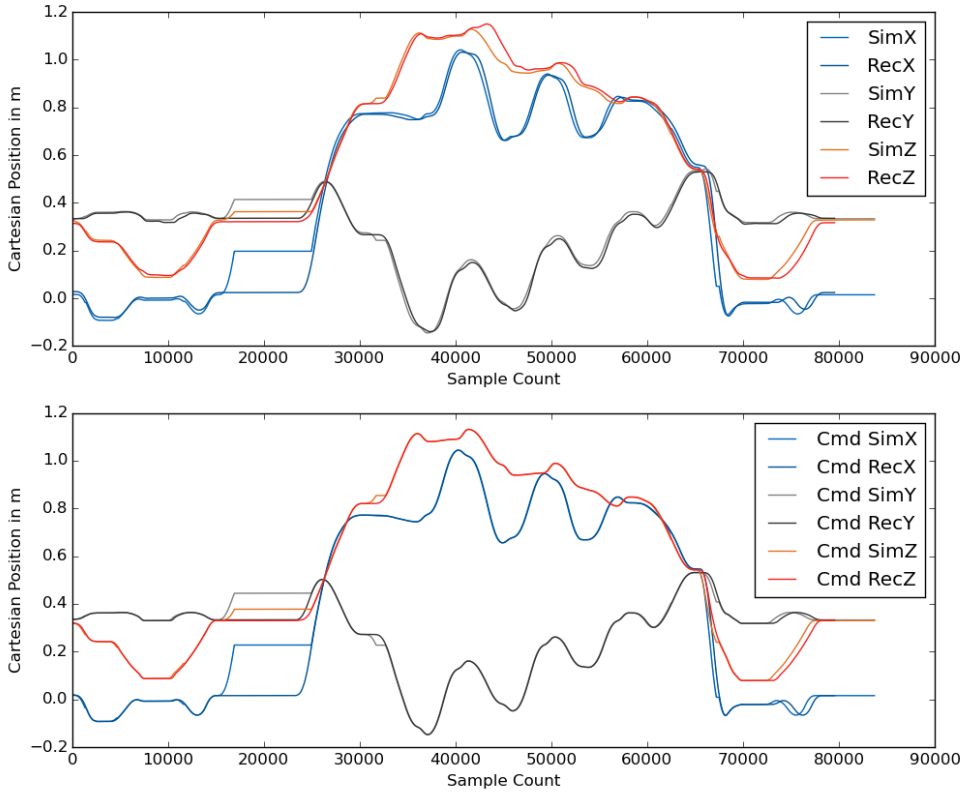
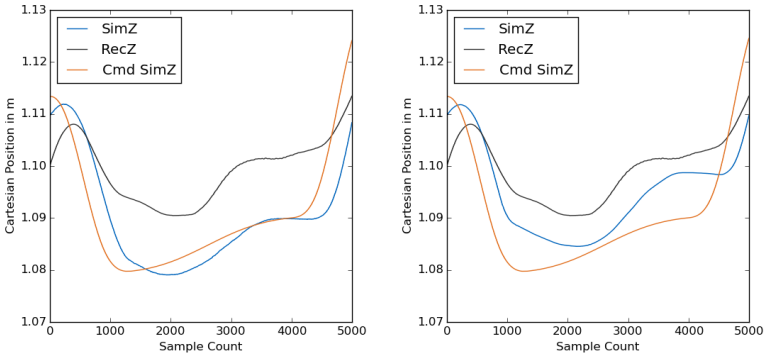


Figure 4.14: Measured (upper diagram) and commanded (lower diagram) cartesian position of the left hand end effector in simulation and reality, divided into the three spatial axes

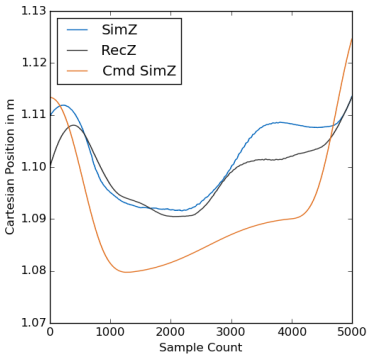
stays in acceptable boundaries.
The developed system was hardly optimized with regards to performance, however the results of performance metrics show that the Gazebo plugin itself leaves little room for improvements.

For robot movements without any collider contacts or objects controlled by physics, the average real-time factor reaches about 0.86, i.e. the simulation time runs with 0.86 times the speed of reality averaged over 10 seconds.
Since surface contacts require solving the torques and positions with more constraints,



(a) Panel height 1.00m

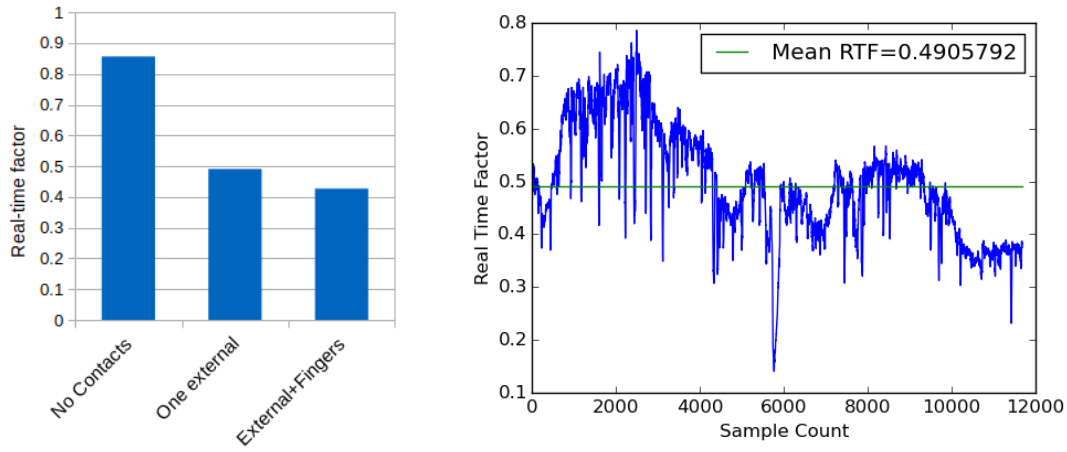
(b) Panel height 1.02m



(c) Panel height 1.04m

Figure 4.15: Z-positions of the simulated end effector for a wiper angle of 0 and varying panel heights in the simulation in comparison to commanded and measured Z-position of the real robot, the low section of the graph in orange indicates the contact phase

the performance drops significantly once Justin's hands interact with any objects in the world. The data used for these measurements were taken from the experiment described in 4.4 where Justin holds a wiper that exerts some force on the surface of a solar panel. For such a singular surface contact, the average real-time factor drops to 0.49, with massive further drops down to 0.14 when the wiper is moved across the surface while in contact with it. This case did appear consistently for the second out of three wiping motions, for the others the drop in performance wasn't as significant, compare Fig. 4.16(b).



(a) Average real-time factor for movements with different contact scenarios

(b) Real-time factor over time for three wiping motions over panel surface with one contact each

Figure 4.16: Real-time factor evaluations of the simulation

When the fingers of Justin also attempt to grip the wiper (which is attached statically to the hand in this scenario) they come into further contact with it. These are multiple contacts that lead to some oscillations in the finger joints, but since these forces are exerted by and onto children of the hand base, they do not influence the arm joints or the rest of the robot.
With the fingers and a panel contact, the real-time factor is at an average of 0.43, a comparison of the cases is visualized in Fig. 4.16(a).
The Gazebo plugin that is responsible for communication with Simulink and for the actuation of the simulated model does not contribute a large amount of runtime for each time step: For any of the scenarios, its average runtime was below 200μs.
The caching of string-based calls to Gazebo's physics interfaces provided good improvements regarding the performance of the plugin. The performance drops in contacts with the solar panel surface could likely be reduced by simplifying the collider mesh.

5 Discussion

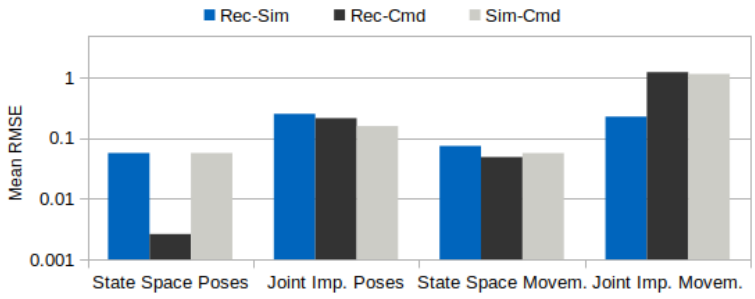
The presented results provide an insight into the accuracy of different controller modes and will be analyzed for possible causes or improvements here. The chapter starts with the body and hand controller and will attempt to draw conclusions on the future use cases of both, and ends with the analysis of the conducted wiping experiment described in section 4.4.

5.1 Controller Accuracy

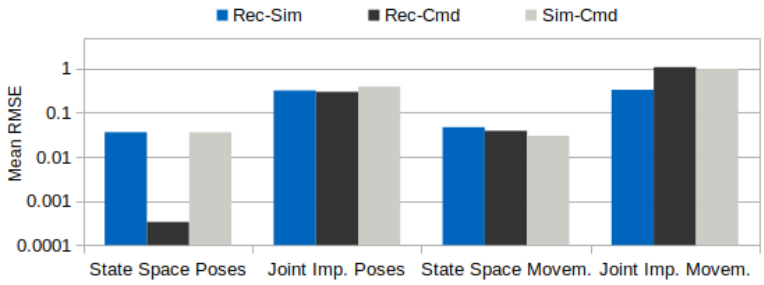
The body controller coupling to Simulink was the focus of this thesis and as such is at the core of future use cases. Its accuracy will be put into relation and an assessment is drawn which limitations and potentials could be found.

5.1.1 Joint Positions

The low maximum position errors in any control mode allow the conclusion that all poses and movements are tracked qualitatively and are constrained to small quantitative errors. The state space controller has significantly higher position accuracy than the joint impedance controller.
Taking a closer look at the diagrams over the time domain also strongly indicates that the state space control mode comes with little to no noise even though the real robot's state space controllers run with 3kHz instead of 1kHz used in simulation.
Obviously even small torso errors can have a strong influence on the position of arms and end effectors since their leverage is quite large, so the very good results for the state space control are very much desirable while the joint impedance control errors should get some attention.
This appears to be an issue with the relative position of the upper body: The movement *torso_back* produces much larger position errors than the park pose although the torso joints deviate far from their default position in both cases. In the first case the upper body starts at a relatively far forward point in relation to the robot base - the upper torso joints experience torques in the same directions. The park position keeps the upper body right above the base, resulting in opposite torques in the two upper torso joints.



(a) Mean RMSE values for the arm joints



(b) Mean RMSE values for the torso joints

Figure 5.1: Mean RMSE between recorded - simulated, commanded - recorded and commanded - simulated positions

A possible conclusion would be that the parallel torso construction has some difficulties holding the upper body when the required torques are large and divided among both upper torso joints.

The joint position accuracy should also be put into perspective by comparing it to the errors between commanded and measured positions on the real robot and in the simulation. The following results are true for arm and torso joints.
A comparison of the simulated positions to the commanded ones shows the state space controller following commanded positions more strictly, just as expected.
The measurements of the real robot show similar discrepancies to the commanded positions except for state space poses.
For joint impedance movements the mean RMSE between simulation and reality is significantly smaller than both discrepancies to commanded positions. This means that the simulated motion tracks the real motion more closely than its own commanded positions, see Fig. 5.1, which is desirable. Generally this shows that the commanded positions are not followed perfectly in reality or simulation and the simulation follows

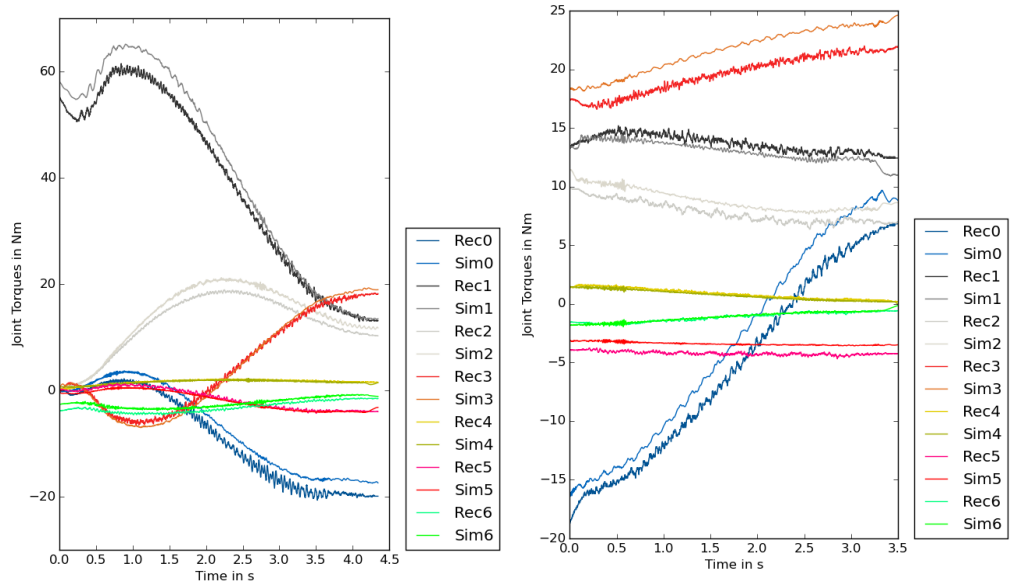
them at a similar accuracy as the real robot, it just doesn't always replicate these discrepancies at the same time or on the same joints (except where the recorded - simulated errors are much lower).
While state space poses are held extremely close to the commanded positions by the real robot compared to the simulation, the simulation errors are still in very small absolute boundaries.

5.1.2 Torques

The torques of the robot also play a role in the position accuracy of the robot, but since they are the controller output used to achieve a certain joint position, the controllers continuously change them according to the requirements, i.e. inaccuracies of kinematic or inertial properties in the model will affect the simulated behavior and therefore the torques required to reach the same joint positions.
This doesn't make them any less important as one of the major goals of this thesis is the achievement of realistic torques and forces in the simulation. The prediction of environment contact forces relies heavily on the accuracy of torques, mainly those of the arm joints for hand manipulation contacts.

We first discuss the torque results for the arm joints.
Generally, the torque errors in the arms are consistently low among both controllers, poses and positions, the mean RMS values remain below 1.5.
The similar torque errors of State Space and Joint Impedance control are almost certainly caused by inaccuracies in the model and the limitations of discrete physics. Both controllers counter the resulting position errors by adapting the torques to slightly different values, creating the torque errors analyzed in this section.
The poses are especially interesting in regard to torques since most compliant contacts with the environment happen with much lower velocities than those of the movements analyzed, e.g. gripping would require the arm movement to be fully finished before interacting with the object. The maximum torque error in poses is below 3.5Nm, which is still larger than the occasional discrepancies observed on the real robot external torques in the wiping experiment examined in section 4.4.2.

Movements showed similar mean RMSE values as poses but much larger maximum RMSE values. The peak of the RMSE at 2.441 in the state space movement from *zero* to *ready* position correlates with a large torque difference necessary to drop the arms all the way to their desired position. The highest single joint torque error at 6.245Nm also occurs in this movement and can be put into relation by looking at the torques around the *zero* position. They reach more than 60Nm at the same joint, reducing it to a relative error of under 10%, see Fig. 5.2(a). This offset is still significant and indicates



(a) Torques in the *zero_to_ready* movement in state space control

(b) Torques in the *ready_to_idle* movement in state space control

Figure 5.2: Separate torque graphs for the right arm joints in simulation and reality

room for improvement e.g. regarding inertial properties, especially considering that the maximum error for holding the zero pose is much lower at 2.73Nm.
This joint is not responsible for the maximum errors of all movements, the comparison of separate torques rather indicates that a certain error relative to the current torque is common among all joints, see Fig. 5.2(b).

The observed errors should be put into relation to the errors between the real robot and the model, i.e. a discrepancy between the commanded torques calculated by the Simulink model and the torques measured by the sensors on the real robot. Normally, the commanded torques are simply applied, but the state space controller gives over the actuator control to joint-level controllers. These aim to reach a commanded position on their joint using a PD controller. The commanded torque from Simulink is therefore not applied in this case.
These commanded torques only consist of the gravity compensation output and do not grow with the position error. This makes a comparison in movements useless, so the error is only calculated for poses in state space control. The joint-level controllers hold the pose against gravity with the necessary torques, which are different than suggested

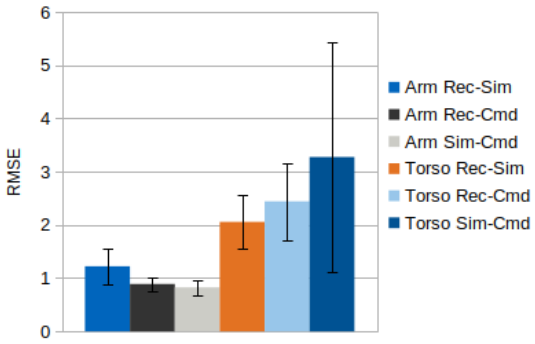


Figure 5.3: Mean RMSE between recorded - simulated, commanded - recorded and commanded - simulated torques

by the Simulink controller.
The mean RMSE between real and commanded torques is at 0.889 for the arms, the RMSE of recorded and simulated torques is at 1.223 and the mean RMSE of simulated and commanded torques is at 0.827, visualized in Fig. 5.3.
This shows that the torque differences between the simulation and the real robot are around the same magnitude as the differences between the assumptions of the Simulink model and the torques required in reality or simulation. For the torso the torque discrepancies between reality and Simulink and between simulation and Simulink are both larger than those between recorded and simulated data: 2.442 between commanded and sensor torques, 3.275 between commanded and simulated torques and 2.057 between sensor and simulated torques. This could suggest that the simulation model is closer to the real Justin than the Simulink model in regard to the torso. However the real and simulated measurements contain gravity compensation torques together with small position error torques which might also be the source of the larger discrepancies to commanded torques.

Regarding the torso, the mean RMSE values are significantly higher than in the arm joints, but remain below 2.5.
The maximum torque errors in the torso are in parts surprisingly large, especially one that is larger than 11Nm for the *torso_back* movement.
Comparing that error to the torque requirements of its movement in Fig. 4.8 shows the relative torque error is at about 10% because the acting torques reach over 110 Nm. The *torso_back* movement also has by far the highest torque changes and the largest cartesian displacement of the upper torso from its default position above the platform. Moving between the standard poses does not produce torque errors larger than 8.5Nm

in the torso.
This larger discrepancy was also found in the torso position errors in the previous section and might be a problem with the parallel torso construction, the specific root of this problem is not clear.

Generally, the usual poses of the robot are held with acceptable torque accuracy in torso and arm joints. Movements between those are only slightly less accurate, while demanding movements that go towards the limits of the real robot can lead to larger torque differences.
The inertial properties of arm joints and the parallel torso construction could be improved in the future, for now the limitations can be respected by avoiding environment contacts while the torso joints experience significant rotations. It's also likely that the torques of torso joints would not play a significant role in the accuracy of external force estimations anyway as long as the forces are applied to arms or hands and not the torso links.

5.1.3 End Effector Pose

The cartesian position error of the end effectors is of high interest for any movement and pose since Justin's hands set the highest demands regarding position and rotation accuracy, larger errors are likely to introduce significant changes to the outcome of tasks, e.g. make the difference between touching a surface and hovering above it.

We focus on the evaluation of simulated positions determined through forward kinematics.
In this case the errors are very small: A mean cartesian error of $0.171cm$ for poses in state space allows for very fine interactions with objects and demonstrates that the remaining joint errors do not accumulate to undesired offsets in the hands.
The fact that the mean cartesian accuracy doesn't decrease significantly in movements is a very desirable effect for environment interactions, but the maximum position error does reach $1.005cm$ in the *zero_to_ready* movement.
The joint impedance accuracy is slightly lower at a mean error of $0.568cm$, caused by the lower joint position accuracy. The errors remain in acceptable boundaries of under $2cm$ which still allows for quite accurate interactions, the mean error of $0.601cm$ in movements can be considered good.
Generally, the end effector position is more reliable in poses than in movements. Fine interactions during such movements are avoided anyway.
The low maximum rotation distance is also a strong indicator for the high precision of the arm joints.

Comparing these results to the errors measured in Gazebo, it's obvious that the position errors of Gazebo are much larger, reaching more than $1cm$ mean cartesian error for both state space and joint impedance control. However, the absolute differences between state space and joint impedance mean errors are similar, at around $0.5cm$. This indicates a constant offset in the measurements taken from the simulation.
The causes for this offset between measured and calculated hand positions are unclear, but likely lie in some accumulated inaccuracies in the Simulink or Gazebo model of Rollin' Justin.
It's also not clear whether the cartesian positions of the real end effectors are calculated completely accurately by the Simulink model. The offset in Gazebo is significant and obviously the Gazebo position is the deciding factor in the outcome of interactions, not the forward kinematics results.

5.1.4 General Assessment

The expectations for the different controllers were fully met - the state space controller provides higher position accuracy for both torso and arm joints, joint impedance control still produces quite accurate results when used with some caution regarding the torso and isn't designed for maximum accuracy on the real robot either.
Overall, the torque accuracy of arm and torso joints does not differ significantly between the two controllers. It's likely that the model could be optimized further regarding its inertial properties to gain small improvements. It's also worth investigating whether adapting the Simulink model makes more sense as the commanded torques from Simulink in state space control are about as far from the real robot's requirements as the simulated torques.
Just like on the real robot, contacts with the environment should happen in a slow and controlled manner to avoid torque errors from movements influencing the contact forces. The larger torque errors in the torso are worth investigating, too, but likely negligible for most scenarios with environment contacts.
The cartesian hand position accuracy fulfilled the expectations regarding the forward kinematics calculations. Interactions with Justin's hands are likely to tolerate the position errors of about $0.17cm$ in state space control even in scenarios with high accuracy demands. Both controllers provide good results that correlate with their joint position accuracy.
The comparison with Gazebo measurements shows an offset that is likely also present in some magnitude on the real robot. It is not trivial to measure Justin's real end effector positions but it would be useful for improving the models used in Simulink and Gazebo.

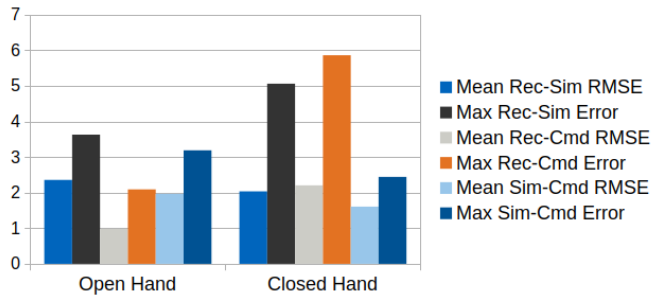


Figure 5.4: Discrepancies of commanded and recorded finger joint positions compared to simulation errors, the maximum errors are in degrees while the RMSE does not have a unit

5.2 Finger Accuracy

The finger positions have much larger errors on average and at their maximum than arm or torso positions for either controller. The graphs of joint positions over time also show a significant error at the start as seen in Fig. 4.11. This diagram does not show the end position of that joint, the graphs are cut off when the commanded angles reach their final point, the real joint does get closer to its commanded position.
A further comparison of the logged positions of the real fingers and their commanded positions demonstrates that the real hand is not following them very strictly. With regard to commanded positions the real fingers execute the open hand pose more accurately, for the closed hand pose the simulation is closer to commanded positions than reality, see Fig. 5.4. The errors between simulated and real fingers are at similar magnitudes as respective discrepancies to commanded positions in both poses.
This puts the simulation errors into relation, but in an ideal case, the simulated finger movements follow the real movements as closely as possible, not the commanded ones. Some parameters could be optimized for this purpose, e.g. the PD gains, inertia tensors as well as friction and damping values.
We chose the parameters for the experiments through a simple grid search approach. Further optimization does not appear necessary since the maximum errors stay in somewhat acceptable boundaries and the Gazebo simulation will not be used for the simulation of gripping. The multitude of collisions and the simplified collider boundaries of the fingers would lead to instability when attempting to hold an object tightly.

5.3 Environment Contacts

The wiping experiment provides insights to the compliant contact behavior of the simulation and demonstrates the usage of parameter variations as a way to deal with world state uncertainty.

The time offsets are caused by additional atomic operations executed between the movements. The shorter observed delays are caused by parameter changes e.g. compensating for the wiper mass. The source for the large delay of about four seconds likely lies in the fact that the wiping action switches to a second action template at that point. The first template is responsible for taking the wiper and the second for wiping the panel and placing the wiper back.
The time discrepancies between simulation and reality are successfully compensated by logging the execution timestamps of the consecutive action steps. This could be further improved by recording the timestamps on the real robot and executing each step in the simulation once the simulation time - not real time - has reached the next timestamp. Live application of the simulation could also solve this problem by sending any user input to two running Simulink controllers - one for the real robot and one for the simulation.

The simulated torques measured on the arm joints are generally of a similar magnitude to those in reality. They do vary significantly with the different parameter sets analyzed in the experiment, which is generally an expected effect and allows for optimization.
The simulated torques appear at the correct times and for similar durations, indicating that the surface contacts are generally replicated well and the wiper is dragged across the surface instead of e.g. getting stuck.
Some of the parameter sets result in no contacts or far too large torques, while others show torques roughly at the desired magnitudes.
However, the projection of the contact force into the arm joints as torques is incorrect in the simulation, most prominently in joints 0 and 3, which experience the largest torques. The torques produced by the simulated panel contact affect the fourth joint much more than the first, in reality this is observed the other way around.
The torque peaks are also more pointed in the simulation. Both of these discrepancies could be caused by the wiper being simulated as a rigid body with just one surface contact point. In reality, the wiper head is a soft body that gets slightly deformed by the contact force and rather creates a line of contact with the panel. This can result in a more evened out spatial and temporal distribution of the contact force and therefore joint torques.

The cartesian end effector position reaches a satisfying accuracy. In a scenario like the wiping experiment the time offsets during movements - while the commanded positions are synchronized - do not have a significant influence on the results because the contacts are far longer than the offsets. Other experiments with higher time accuracy demands could run into difficulties. It would be worth investigating whether this behavior occurs only in contact scenarios or generally in cartesian impedance control.
The low errors on the x and y axis also confirm that the wiper is dragged across the surface at a correct speed in the simulation. This indicates that the friction force acting against the movement direction has a similar effect on the movement as in reality, even though the real wiper would have different physical properties with its soft head.
The z-position errors shown in the results demonstrate a limit of the rigid body simulation: A z-position difference of half a centimeter significantly changes the torques and a more accurate z-position in the simulation comes with unrealistically large torques. A certain trade-off is unavoidable in this case.
A closer analysis also shows slightly higher accuracy of x- and y-positions for the largest panel height with the wiper angle of 0.0. This indicates that this parameter combination represents the real world state the best. The much too large dimensions of the torques occurring in that case might be due to the compression of the soft wiper head in reality that is not replicated in the simulation and that would reduce the contact force.

The parameter variations for this experiment did not result in a single ideal parameter set but rather give boundaries to acceptable parameter combinations since some of them resulted in no contacts or excessively hard contacts. The combinations of a wiper x-angle of 0.0 and a panel height of 1.00m, a wiper x-angle of -0.075 rad and a panel height of 1.02m and a wiper x-angle of -0.15 rad with a panel height of 1.04m all resulted in similar torques that were at roughly the same magnitudes as the real values. The best position accuracy occurred in the combination of a panel height of 1.04m with an x-angle of 0.0, but also resulted in joint torques several times larger than the ones measured in reality.
The integration of a soft body simulation or an estimation with a spring at the head of the wiper could likely increase the simulation accuracy for this wiper application.

It's clear that the choice of two parameters is not sufficient to deal with the uncertainty of the wiper pose and the panel position, a total of ten parameters would be available for the two objects, one rotation and three position variables for the panel and 3 position plus 3 rotation variables for the wiper pose relative to the wrist of the left arm.
We inserted some knowledge manually, like the fact that the robot and panel can only be rotated along their vertical axes even though the visually determined world state

suggests otherwise. We also manually determined the wiper pose and chose one of the angles as the most significant regarding surface contacts.
Future experiments could attempt to include more of the available variables or find ways to choose the significant ones with a data-based approach. The wiper pose could also be estimated initially by simulating the grasping behavior in a separate hand simulation or with a dynamic monitoring that binds the wiper to the hand when the fingers close around it.
The discrepancy in the proportions of the projection of contact forces onto arm joint torques stood out negatively and should be looked into in the future.
A simpler experiment that might grant more insight to the current compliant contact behavior would be the contact of Justin's hand with the surface of a scale. This would allow us to get the contact force directly, would not require an end effector tool and could be varied with the desired end effector position at different depths below the scale surface. In Gazebo, the direct access to surface contact forces is possible with contact sensors.

6 Conclusion and Outlook

We draw a conclusion of the coupled systems, the experiment results and discussion and provide an outlook on the use cases and possible improvements to the system.

6.1 Conclusion

The coupling process of the Simulink controller with a simulation was generally successful and the processes are synchronized reliably. We explained improvements and limitations of the robot model with a parallelogram torso construction as the largest change. The simulation is now fully driven by torques and forces rather than kinematic positions of robot elements.
We also conducted a deeper accuracy analysis of the simulated controller behavior regarding joint angles, torques and the end effector positions and orientations. The joint and end effector positions were replicated with satisfying accuracy in both state space and joint impedance control, the measured torques showed some discrepancies, especially in the torso.
We implemented a separate simple finger controller that provided acceptable results, consistent contact-based grasping would need to be simulated separately anyway.

With the wiping task, we successfully simulated a typical action of the real Justin including compliant contacts with a flat surface. Parameter variations of the end effector tool and the solar panel position demonstrated a way to deal with uncertainty regarding the world state. The grid search approach lead to a number of superior parameter combinations without a clear ideal one.
The end effector positions were replicated well even during the surface contacts, but some limits of the simulation were demonstrated by the varying torque accuracy.
Generally the impedance control approach clearly replicates compliant contacts in the simulation and allows for control over the contact forces.
These results allow for the conclusion that the main objective of this thesis is fulfilled, the simulation can now replicate force-based applications of the robot. The conduction of further experiments could grant more insights to the current compliant contact behavior and could help improve and define limits to the accuracy of the simulation e.g. regarding soft bodies or collider geometry.

6.2 Outlook

The developed coupled simulation can fulfil any of the purposes of the previous simulation and extends the range of applications:
Kinematic collision checks or initiation of physics-based interactions of other objects - such as dropping a ball into a container - are more accurate when the robot's joint positions follow controller input than when they replicate the commanded poses. Collisions can be either detected through the physics engine API or using the metrics explored in 4.4.
The metric of a contact force was previously useless as the kinematic joint commands produced undefined torques, these are now realistically simulated and could e.g. be used in the world state interpretation, as training data for contact classification or in the predictive abilities.
The projection methods can now include torques, forces and the compliant controller behavior which enables more advanced scenarios like the panel wiping explored in this thesis, the connection of a plug to a socket or interactions with heavier objects where the simulation could help estimating load limits of the robot.
To accommodate for uncertainties in the estimated world state, multiple simulations with slightly varied poses or parameters - as demonstrated in the wiping experiment - can be executed during or preceding the real robot action.
Future work could develop more sophisticated approaches to dealing with this uncertainty e.g. by calculating probabilistic success chances of planned actions and intervening when large contact forces are risked.
These use cases can be implemented with little effort, building scenarios in the simulation only requires 3D models and their world positions. Many of the systems working on the real Justin - action planning, user interaction methods - could be plugged into the simulation as they mostly interact with the Simulink controller. Some require additional input like a camera input for localization, which would require extensions of the simulation.
The controller itself can also be easily tested, the changes required to interface with LN are minar and can be matter in a matter of minutes. Compiling a new controller and deploying it to the simulation is done locally and quickly, there are no hardware faults like on the real robot and errors in the controller programming can't have negative consequences.
The robot model can also be grounds for testing: Simulating sensor or actuator defects can help identify possible outcomes or detect and localize failures on the real robot faster.

Some of the shortcomings of the coupled simulation such as the cartesian position

offsets can be tracked down to the discrepancies between the model used in Gazebo, the real robot and the model in Simulink. A unified model of the robot's links and joints including their inertial properties should be an easy way to improve the simulation and can be achieved partly through automatic evaluation of CAD files of robot parts.
The simulation of gripping objects with multiple fingers is difficult to achieve due to the multitude of contact forces, shifting this logic to a different simulation that only deals with finger contacts may be worth exploring and could integrate the Simulink controller of Justin's hands instead of the PD controller implemented in Gazebo.
The rolling platform of the robot that wasn't handled in this thesis could also be implemented in future work, it was fixed to the origin in the current simulation. The four wheel-contacts to the ground might pose a challenge to the stability of the robot while the result hardly contributes to the simulation use cases. Instead moving the platform according to the interpolated commanded movements on two prismatic joints along the x and y axis of the world using forces would likely be recommended and could be realized with much less effort.

List of Figures

List of Tables

Bibliography

[1] N. Y. Lii, D. Leidner, A. Schiele, P. Birkenkampf, B. Pleintinger, and R. Bayer, "Command robots from orbit with supervised autonomy: An introduction to the meteron supvis-justin experiment," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts*, ser. HRI'15 Extended Abstracts, Portland, Oregon, USA: Association for Computing Machinery, 2015, pp. 53–54, ISBN: 9781450333184. DOI: 10.1145/2701973.2702022.

[2] A. S. Bauer, P. Schmaus, F. Stulp, and D. Leidner, "Probabilistic effect prediction through semantic augmentation and physical simulation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 9278–9284. DOI: 10.1109/ICRA40945.2020.9197477.

[3] D. Leidner, C. Borst, A. Dietrich, M. Beetz, and A. Albu-Schäffer, "Classifying compliant manipulation tasks for automated planning in robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015.

[4] D. Leidner and M. Beetz, "Inferring the effects of wiping motions based on haptic perception," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 461–468. DOI: 10.1109/HUMANOIDS.2016.7803316.

[5] M. A. Oliveira, S. Doncieux, J.-B. Mouret, and C. Peixoto Santos, "Optimization of Humanoid Walking Controller: Crossing the Reality Gap," in *IEEE RAS International Conference on Humanoid Robots (Humanoids 2013)*, Atlanta, GA, United States, Oct. 2013, pp. 106–111. DOI: 10.1109/HUMANOIDS.2013.7029963.

[6] S. Koos, J.-B. Mouret, and S. Doncieux, "Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers," in *Conference on Genetic and Evolutionary Computation*, United States: ACM, publisher, Jul. 2010, pp. 119–126.

[7] J. Collins, D. Howard, and J. Leitner, "Quantifying the reality gap in robotic manipulation tasks," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6706–6712. DOI: 10.1109/ICRA.2019.8793591.

[8] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, eaau5872, 2019. DOI: 10.1126/scirobotics.aau5872. eprint: https://www.science.org/doi/pdf/10.1126/scirobotics.aau5872.

[9] S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, M. Mozifian, F. Golemo, C. Atkeson, D. Fox, K. Goldberg, J. Leonard, C. Karen Liu, J. Peters, S. Song, P. Welinder, and M. White, "Sim2real in robotics and automation: Applications and challenges," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 2, pp. 398–400, 2021. DOI: 10.1109/TASE.2021.3064065.

[10] E. Salvato, G. Fenu, E. Medvet, and F. A. Pellegrino, "Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning," *IEEE Access*, vol. 9, pp. 153 171–153 187, 2021. DOI: 10.1109/ACCESS.2021.3126658.

[11] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "Bigdog, the rough-terrain quadruped robot," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10 822–10 825, 2008, 17th IFAC World Congress, ISSN: 1474-6670. DOI: https://doi.org/10.3182/20080706-5-KR-1001.01833.

[12] C. E. Agüero, N. Koenig, I. Chen, H. Boyer, S. Peters, J. Hsu, B. Gerkey, S. Paepcke, J. L. Rivero, J. Manzo, E. Krotkov, and G. Pratt, "Inside the virtual robotics challenge: Simulating real-time robotic disaster response," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 494–506, 2015. DOI: 10.1109/TASE.2014.2368997.

[13] E. Coevoet, A. Escande, and C. Duriez, "Soft robots locomotion and manipulation control using fem simulation and quadratic programming," *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, pp. 739–745, 2019.

[14] L. Sentis, J. Park, and O. Khatib, "Compliant control of multicontact and center-of-mass behaviors in humanoid robots," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 483–501, 2010. DOI: 10.1109/TRO.2010.2043757.

[15] C. Semeraro, M. Lezoche, H. Panetto, and M. Dassisti, "Digital twin paradigm: A systematic literature review," *Computers in Industry*, vol. 130, p. 103 469, 2021, ISSN: 0166-3615. DOI: https://doi.org/10.1016/j.compind.2021.103469.

[16] F. Biesinger and M. Weyrich, "The facets of digital twins in production and the automotive industry," in *2019 23rd International Conference on Mechatronics Technology (ICMT)*, 2019, pp. 1–6. DOI: 10.1109/ICMECT.2019.8932101.

[17] K. S. D. Ravi, M. S. Ng, J. Ibáñez, and D. Hall, "Real-time digital twin of on-site robotic construction processes in mixed reality," Nov. 2021. DOI: 10.22260/ISARC2021/0062.

[18] M. Kapteyn, D. Knezevic, D. Huynh, M. Tran, and K. Willcox, "Data-driven physics-based digital twins via a library of component-based reduced-order models," *International Journal for Numerical Methods in Engineering*, vol. n/a, no. n/a, DOI: https://doi.org/10.1002/nme.6423. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.6423.

[19] F. Laukotka, M. Hanna, and D. Krause, "Digital twins of product families in aviation based on an mbse-assisted approach," *Procedia CIRP*, vol. 100, pp. 684–689, 2021, 31st CIRP Design Conference 2021 (CIRP Design 2021), ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2021.05.144.

[20] H. Meyer, J. Zimdahl, A. Kamtsiuris, R. Meissner, F. Raddatz, S. Haufe, and M. Bäßler, "Development of a digital twin for aviation research," in *Deutscher Luft- und Raumfahrt Kongress*, Sep. 2020.

[21] E. Glaessgen and D. Stargel, "The digital twin paradigm for future nasa and u.s. air force vehicles," in *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*. DOI: 10.2514/6.2012-1818. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2012-1818.

[22] K. Hagmann, A. Hellings, J. Klodmann, R. Richter, F. Stulp, and D. Leidner, "A digital twin approach for contextual assistance for surgeons during surgical robotics training," *Frontiers in Robotics and AI*, no. 8, pp. 305–319, Sep. 2021.

[23] A. Albu-Schäffer and G. Hirzinger, "A globally stable state feedback controller for flexible joint robots," *Advanced Robotics*, vol. 15, no. 8, pp. 799–814, 2001. DOI: 10.1163/156855301317198133. eprint: https://doi.org/10.1163/156855301317198133.

[24] C. Ott, O. Eiberger, W. Friedl, B. Bauml, U. Hillenbrand, C. Borst, A. Albu-Schaffer, B. Brunner, H. Hirschmuller, S. Kielhofer, R. Konietschke, M. Suppa, T. Wimbock, F. Zacharias, and G. Hirzinger, "A humanoid two-arm system for dexterous manipulation," in *2006 6th IEEE-RAS International Conference on Humanoid Robots*, 2006, pp. 276–283. DOI: 10.1109/ICHR.2006.321397.

[25] A. S. Bauer, A. Köpken, and D. Leidner, "Multi-agent heterogeneous digital twin framework with dynamic responsibility allocation for complex task simulation," in *21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022)*, P. Faliszewski, V. Mascardi, C. Pelachaud, and M. E. Taylor, Eds., IFAAMAS, May 2022, pp. 1–9.

[26] T. Chai and R. Draxler, "Root mean square error (rmse) or mean absolute error (mae)?– arguments against avoiding rmse in the literature," *Geoscientific Model Development*, vol. 7, pp. 1247–1250, Jun. 2014. DOI: 10.5194/gmd-7-1247-2014.

[27] D. Huynh, "Metrics for 3d rotations: Comparison and analysis," *Journal of Mathematical Imaging and Vision*, vol. 35, pp. 155–164, Oct. 2009. DOI: 10.1007/s10851-009-0161-2.

[28] D. Leidner, C. Borst, and G. Hirzinger, "Things are made for what they are: Solving manipulation tasks by using functional object classes," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, 2012, pp. 429–435. DOI: 10.1109/HUMANOIDS.2012.6651555.