Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

BACHELORTHESIS

Evaluating and Minimizing the Reality Gap in the Domain of RoboCup Humanoid Soccer

vorgelegt von

Tanja Flemming

MIN-Fakultät

Fachbereich Informatik

Arbeitsbereich Technische Aspekte Multimodaler Systeme

Studiengang: Computing in Science

Matrikelnummer: 6774486

Erstgutachter: Dr. Norman Hendrich

Zweitgutachter: M. Sc. Marc Bestmann

Abstract

The Reality Gap describes the difference of motion behavior of objects, e.g., robots, between simulation and reality. In this thesis, an approach to minimize it is presented. The minimization of the Reality Gap is important to be able to use the simulation meaningful. Different approaches to minimize it were studied in the literature. It was decided to improve the simulation model of the robot in this thesis. The different improvements on the model are presented and evaluated regarding their success in minimizing the Reality Gap. The used methods were successful in minimizing the Reality Gap of the joints simulated position and torque. Furthermore, more in-depth knowledge of the robots motors behavior is gained, which can be helpful in the future to improve the velocity simulation accuracy and to further improve the position and torque accuracy. This method can be applied to other robots as well to minimize but not close the Reality Gap.

Zusammenfassung

In dieser Bachelorarbeit wird ein Ansatz zur Minimierung der Reality Gap untersucht. Bei der Reality Gap handelt es sich um den Unterschied im Bewegungsverhalten von Objekten, wie Robotern, zwischen Simulation und Realität. Dies zu untersuchen ist wichtig, um Simulation sinnvoll nutzen zu können. In dieser Arbeit werden verschiedene Ansätze zur Minimierung der Reality Gap untersucht. Umgesetzt wurden Verbesserungen am Simulationsmodel des Roboters, welche danach evaluiert wurden. Die umgesetzten Verbesserungen konnten im Bezug auf die Simulation der Position und des Drehmomentes der Gelenke des Roboters die Reality Gap erfolgreich minimieren. Außerdem ist ein tieferes Verständnis der in dem Roboter in den Gelenken verbauten Motoren erlangt worden. Dies ist nützlich für zukünftige Verbesserungen der Simulation der Geschwindigkeit der Robotergelenke, sowie der Position und des Drehmomentes. Die in dieser Arbeit verwendete Methode ist auf andere Roboter anwendbar, um die Reality Gap zu minimieren, jedoch nicht um sie zu schließen.

Contents

List of Figures

List of Tables

Acronyms

AI artificial intelligence
CAD computer-aided design
DC direct current
DoF degrees of freedom
FIFA Fédération Internationale de Football Association
IMU inertial measurement unit
ODE Open Dynamics Engine
PID proportional–integral–derivative
PLA polylactic acid
SDF Simulation Description Format
ROS Robot Operating System
UAV unmanned aerial vehicle
URDF Unified Robot Description Format
XML Extensible Markup Language

1. Introduction

In robotics, simulation is used as a tool for various applications because it protects the hardware when, e.g., the software is tested, or a robot's motion is learned. It also can be used outside of the lab, and it saves time if the software or new hardware designs are tested. All these applications have to deal with the so-called Reality Gap, which describes the phenomena that in simulation produced results do not work completely equal when used in reality. The Reality Gap is the difference between a robot's motion behavior in simulation and reality. Therefore a direct transfer from results developed in simulation to a real robot is often not possible. This difference is mainly based on the problem that simulators are not able to replicate the physics of the environment precisely. Another problem constitutes of robot description files, which often contain not the precise parameters to describe the robot. Hence it is currently not possible to model the robots actions and its interaction with the environment realistically. Despite these problems, simulation has the advantage of protecting the, in most cases, expensive hardware, especially when machine learning is used for which multiple possibly damaging training runs have to be conducted. For machine learning, this is also faster because the learning can be distributed over multiple systems. The importance of minimizing the Reality Gap is illustrated in Section 1.1. The goal of this thesis is described in Section 1.2, and in Section 1.3, the outline of the thesis is presented.

1.1. Motivation

Closing the Reality Gap is essential for many research problems according to a user study [18] in the robotics community. In that study, it was figured out, that the most important criterion for choosing a simulator is that it is as close as possible to reality. Another significant criterion mentioned is that the same code can be used for the real and the simulated robot.

Particularly in RoboCup Soccer closing the Reality Gap is of great interest. RoboCup [O26] is a competition in robotics to support research in this area. It is structured in several leagues, e.g., the RoboCup soccer leagues. The soccer leagues are divided into the ones working with a humanoid robot, leagues using wheel driven robots and simulation leagues. The soccer leagues aim to win against the FIFA human soccer world champion in 2050 [O25]. For achieving this goal, it will be necessary to join the different leagues in the future. Therefore, the achievements of the simulation leagues have to be transferred to the real humanoid robots, which would work more smoothly

with realistic simulation [26].

An example where a more accurate simulation would be helpful is a humanoid robot, who learned in simulation to kick a ball over a certain distance [1]. The thereby learned kick works in simulation, but the execution of the kick, in reality, is not possible because the final position is the robot standing on its tiptoes. In an accurate simulation, this kick would not work as well and the algorithm would create results usable on a real robot.

Another example is described in [46]. The RoboCup humanoid soccer team NUbots optimized their walking and kicking movement in simulation with genetic algorithms. While achieving working results in simulation with their approach, they encountered several problems when they tried using the movements on the real robot. So the optimized movements were not usable quickly. They explain it with an imprecise simulation. The main effects are the servos and the friction between the robot's feet and the ground. With a more accurate simulation, these problems could be overcome, and the optimization method could be used for the real robot.

An improvement of the simulation accuracy for the robot used in this thesis is helpful for the RoboCup humanoid soccer team Hamburg Bit-Bots, who uses this robot. For example, tuning the parameters of the kick or the walking with machine learning approaches in the simulation would save lots of time, which is needed to tune the parameters by hand. Furthermore, testing in the simulation could then be used more efficiently. It would be possible to perform continuous integration testing of the full software stack of the Hamburg Bit-Bots. So it could be recognized fast if a bug was introduced in the software, which keeps the robot from executing the basic soccer functionality, e.g., scoring a goal on an empty field. Furthermore, it is interesting to know how accurate the simulation of the used robot is to be able to decide how simulation can be purposefully used. Thus it is necessary to evaluate and minimize the Reality Gap to be able to use simulation efficiently without having troubles using the results on real robots.

1.2. Thesis Goal

As it is crucial to be able to use the simulation more efficiently, the simulation has to get more accurate. A first step is achieved in this thesis by improving the description model of Wolfgang robot platform. The Wolfgang robot platform is used as a humanoid robot playing soccer autonomously in the RoboCup Humanoid Soccer League. Therefore improving the description of the robot will help to be able to use results from the simulation soccer leagues and using simulation as a more powerful tool for further developments. The improvements will lead to a more realistic simulation as then the physics engine can work with parameters describing the robot correctly. Even if the physics engine would be perfect, it can not work correctly if the model of the robot is not describing the robot as it is constructed in reality. Furthermore, it will be evaluated how big the Reality Gap of the used robot is, to be able to decide if it is useful to use the simulation under

the existing conditions, or if more work has to be done.

1.3. Thesis Outline

After an overview of approaches to minimize the Reality Gap is given in Chapter 2, the basics necessary to understand this thesis are explained in Chapter 3. This chapter consists of an introduction of the used software and hardware. Additionally, an explanation of the physical background, this thesis is based on, is given. The next chapter (4) introduces the approach taken to minimize the Reality Gap. The improvements are evaluated in Chapter 5. In Chapter 6, the results are discussed, and the state of the simulation of the used robot is assessed. In the end, in Chapter 7, a conclusion is drawn, and areas of future work are illustrated.

2. Related Work

This chapter gives an insight on work by other researchers, which was conducted to minimize or evaluate the Reality Gap. First, some examples are given for which kinds of robots and purposes this work is done. Second, not yet mentioned approaches are explained divided by what is studied to achieve the goal of bridging the Reality Gap. At the end of the chapter, a summary is given.

The Reality Gap is studied due to problems while transferring control algorithms acquired in simulation to the real system. This phenomenon is studied for different robots. For example, in the Deep Mimic Project [31], a neural network was trained to mimic motions. Among other motions, the network learned with reinforcement learning how a simulated humanoid model could realistically run in simulation. For learning the running movement, a reference motion from a real human was used. By adding the reference motion to the training, a realistic running movement was achieved for the model. Before that, without the usage of a reference motion, the training produced an odd behavior. That training resulted in the model running with its arms up and an unnatural leg movement. So the developed training method can be employed to make the simulation more realistic and can, therefore, be used to close the Reality Gap further.

Another example is the simulation of unmanned aerial vehicles (UAVs) to protect the hardware. Meyer et al. describe how a simulation setup in ROS and Gazebo (see Sections 3.2.1 and 3.2.2 for more information about the software tools) was built for the UAVs [29]. For a realization of realistic behavior of the UAVs, two plugins were written for Gazebo to create a dynamic model which takes the flight and motor dynamics into account while being affected by aerodynamic effects, e.g., the wind. For the sensor simulation, available Gazebo plugins were used. For validating the dynamic simulation model, a test trajectory had to be performed by a UAV in simulation and reality. With the development of the plugins, the Reality Gap was reduced as for this robot type, the simulation is now more realistically.

For industrial robots, closing the Reality Gap is also researched. Golemo et al. improved the simulation of a robot arm [17]. For this, a recurrent neural network was trained on the differences of the arm's trajectories of the real and simulated robot. With this neural network, the simulation then was improved, so that policies learned in the simulation are now more easily transferable to the real robot, which represents a minimization of the Reality Gap.

A further example can be found in the domain of space robotics. As described by Yang et al., it is necessary to test robots in simulation before they are launched to space [43].

So a simulation method to validate the robots dynamic characteristics of a free-floating manipulator gripping a satellite was developed. The validation of motion planning is composed of a real hardware experiment to get the joint torques as input for the simulation, which is afterward used to reproduce the forward dynamic characteristics in space. So with this hybrid simulation setup, the reality can be modeled more precisely.

Overall the Reality Gap can be studied for every simulation setup, which is influenced by dynamics. An example is the investigation of a modified version of a children's game [3]. The game consists of a labyrinth, in which a marble has to be navigated to a goal by balancing the game board. Additionally, as difficulty, the game board contains holes in which the marble must not fall. In this study, the game board was controlled by two servo motors. The whole system was modeled in the simulator MARS [O18]. For closing the Reality Gap in this scenario, the noise of the potentiometer for measuring the game boards' deflection, the positioning accuracy of the servo motors, and the measurement uncertainty of the marble detection by a camera were studied. Furthermore, the motor's parameters of the simulation were optimized to get a more accurate model.

As shown above, there exist multiple approaches to close the Reality Gap. Generally, the approaches can be grouped into several categories. One category is the improvement of the actuator simulation, which for example covers the development of equations to describe the motor's properties more accurately, or to implement a motor model for a simulator. Another category involves the improvement of the dynamic simulation model used to describe the robot's kinematics for the simulator. A further category considers the simulation of sensors and the noise acting on them. The next category is the enhancement of the used physics engine by, for example, adding a new feature to make the simulation more accurate or testing which engine is the most appropriate for the considered application. Furthermore, one category is the usage of machine learning or other AI techniques. These are, for example, applied to learn differences between the reality and the simulation to improve the latter, or to combine training data from simulation and reality in a particular procedure to bridge the Reality Gap. Each category is presented in the following, and at the end, a summary is given.

Actuator Model

In the area of improving the actuator simulation, one approach is to use actuator models provided by the simulator and parameterize them correctly. For example, Nguyenová et al. built a dynamic model in the V-REP simulator [O35] for a hexapod robot [30]. For modeling the servo motor, a revolute actuator model provided by V-REP was used. The authors state that the model parameters having the highest effect on the robot's motion are the motion controller parameters and friction parameters. Therefore a PID controller module from V-REP was parameterized with the values used by the real robot and for the friction the related material properties were set. The other parameters affecting the motor in reality, e.g., the gear ratio, were neglected. The evaluation of the built model conducted by comparing the motor positions from a movement executed on the real

robot and in simulation.

A similar example is presented by Denisov et al., which illustrates the creation of a humanoid robot's simulation model with Simulink [13, O36]. First, the model itself was built while taking into account the inertial characteristics of the masses. Second, the actuators were connected to these model by adding the model of a servo motor provided by Simulink. As there was no module for interactions between the robot and the ground, a model to simulate the contact forces was built by connecting provided Simulink blocks. Because some used blocks are approximations, the authors declare that improvements have to be conducted to bridge the Reality Gap. However, that simulation it is already useful for the design and engineering of humanoid robots.

Zhang et al. pursued the approach to use the provided motion controller to simulate a hexapod robot in the Virtools simulator and dynamically parameterize the torque factor of the motion controller [47]. The authors state that the torque factor depends on the distance the motor has to cover and the turning direction of the motor. They worked out a formula for both turning directions of the servo motor which describes the influence of the covered distance to the torque factor. This was then used to dynamically set the torque factor of the motion controller while the robot is moving in simulation.

If the used simulator does not already provide the actuator model, it has to be implemented. An actuator model for a humanoid robot is, for example, missing in the SimTwo simulator [10], for which a servo motor model was developed [24]. This development was focused on modeling the DC motor. The parameters needed for this model were measured at the motor in reality. Furthermore, a simple friction model consisting of a parameter for the static and one for the dynamic friction was implemented. The two friction parameters were found by minimizing the difference of the joint angles when a simple designated task is performed in reality and simulation. Later on, this servo model was improved by adding a gearbox model, which takes the rotor inertia and gear ratio into account [23].

An analysis of a DC Motor and a model implementation for simulation were also conducted by Mensink [28]. Therefore a motor was reverse engineered and the internal friction, as well as the torque constant, was measured. For the simulation model, a setpoint generator was developed to handle the motor's inputs correctly. Additionally, a PID controller was implemented, and a model of the motor's electronics was developed using the results from the experiments with the real motor. For evaluating the built motor model, the position, velocity, and torque from the simulated motor were compared with a real one.

A further study [21] deals with the friction and gearbox deformation in servo motors and models these physical aspects in simulation. For modeling the friction, the LuGre friction model [25] was applied and for the deformation, a generalized Maxwell model [O9]. In reality, measurable parameters for these models were measured at a motor. The remaining parameters were identified with an evolutionary approach based on multiple test scenarios. With these results, a servo motor model in the simulation environment

Breve [O2] was developed. The author states that the model is physically still incomplete but already shows a realistic behavior.

Another approach to improve the simulation's actuator model is the development of equations regarding effects acting on a real motor, which were neglected so far in the simulation models. For example, the friction model of the robot's joints was extended [5]. The effect of joint angles, load torque, and temperature on static friction was investigated at the example of an industrial robot's arm. It was figured out that the joint angle and perpendicular torque do not have a high effect on the static friction, while temperature and manipulation torque affect the static friction. With theses results, the static friction can be described as a function of the motors angular velocity, the joint angle, the perpendicular torque, the manipulation torque, and temperature.

Also, the internal effects of the motor itself have been studied. An example is the study of the effects of gear reduction and the integration of that in the equations describing a robot model with joints [7]. Especially the effects of the motor's rotor inertia and gear ratio were studied. In the paper, an instruction is given how one can adapt the motion equations of a robot by adapting the inertia matrix to calculate the active forces regarding these effects as well. A different approach to this topic is the integration of the motor's inertias and interactions between the links and motors in a dynamic model [36]. The developed, dynamic model realizes a linear parametrization.

Robot Model

Besides tuning the parameters of the actuator, the robot's model parameters also can be improved. This improvement is a step, which has to be taken for using a robot in the simulation. How the model is created regarding its mass and inertial link parameters is for example, shortly described by Denisov [13]. In the context of ROS usage, it is described by Jie et al., how a robot description file can be created [39]. More information about the description file type can be found in Section 3.2.3. For determining the needed parameters for the robot description file, the considered robot was disassembled. Then the links were measured, weighed, and the center of mass was determined. Moreover, the inertia was estimated regarding the shape and mass of the links. Thereby the density distribution was not taken into account. For simplifying the determination of the inertia parameters, the project ROSdyn [O31] by ROS industrial was started to create a ROS package which can autonomously calibrate the dynamic model of a robot.

Validation of a built robot's model in simulation is important, which is shown by Claes [9]. In that study, the simulation of the modeled robot got unstable due to the small size of the robot, which led to numerical instabilities. For achieving a numerically stable simulation, all parameters were proportionally upscaled. The validation was done by comparing a walking motion from reality and simulation.

Sensor Model

For simulating a robot's behavior completely, the sensors have to be included in the simulation. A sensor model has to be defined, which simulates the real sensor, to achieve this. The simulation of an IMU is described by McAnanama and Marsden [27]. As the simulated IMU is developed for UAVs the input consists of position, velocity, and flight altitude. Out of this, the IMU's output is calculated. The sensor's model thereby takes the noise occurring on a real IMU into account by adding an error model to the calculation. This error model is composed of deterministic noise, which is, for example, arising due to temperature impacts and stochastic noise for changing influences.

Yang et al. tested the sensor model of an IMU and observed that the impact of the vibration of the used UAV leads to significant differences in the output from the simulated and the real sensor [44]. So issues still have to be solved to get an accurate sensor simulation for IMUs and other sensors used in robots, e.g., foot-pressure sensors.

Physic Engines

Besides improving the simulation parameters or adapting or extending the simulation software, it is also possible to work with the physics engine itself. One approach for this is to evaluate the physics engine used by the simulator [41]. For accomplishment, a robot had to perform an easy task, which focuses on various physical properties. In this case, the robot pushed an object over a table. The evaluation is done by comparing the final position of the object. Furthermore, a way to optimize the physical parameters of the engine is shown. With this, it can be evaluated which physical engine models the dynamics of the considered robot best, and the usage of the engine can be tuned through correct parametrization.

Another approach for evaluating a physics engine is presented by Drumwright et al. [14]. In this study, problems of the physics engine ODE were identified, while working on mobile robot locomotion and a grasping task of a manipulator. So they reimplemented the parts where they identified problems, to reduce the difference between simulation and reality. The improvements are shown through comparison in multiple robotics-related scenarios.

AI Usability

Direct transferability of learned policies is not possible due to the Reality Gap. Therefore existing approaches to transfer in simulation learned motions are explained further. Besides the one mentioned at the beginning of this chapter, in which a neural network was trained on the differences between an executed motion in simulation and reality to improve the simulation with that, one approach consists of developing a realistic robot and actuator model as a basis [39]. With this, a robust controller should be learned

which can deal with small model errors. The development of this robust controller is achieved by using randomizing dynamic parameters, adding random perturbations, and using a compact observation space.

A further approach focuses on calibrating the simulation parameters, so the robot can learn a trajectory along a reference trajectory, which then is executable on the real robot [38]. For this, it was focused on adapting the actuator parameters and the center of mass positions. While these parameters are essential to simulate the robot's dynamics more accurately, in this approach, the learned parameters do not imitate the real ones. They just fit for this one trajectory, for which they have been calibrated.

It was also tried to optimize the simulation parameters with real-world data [32]. The studied parameters were friction, damping, speed, torque, and other motor-related values. Furthermore, noise is added to the simulator's predictions. The authors state that it is necessary for transferability of the controller to have accurate simulation parameters, while the addition of noise is just helpful to bridge the Reality Gap. Further, they declare that more parameters need to be studied to make the simulation more realistic.

Another approach consists of training on a distribution of simulated scenarios [6]. For this, simulation randomization with the usage of real-world data is conducted to find several parameter distributions for learning in simulation. The goal is to obtain some parameter distributions, which altogether produce a learned policy usable on the real robot.

RoboCup

In the context of RoboCup, the Nao Team Humboldt tried to use the same code for the Standard Platform League and 3D Simulation League [42]. Both leagues use the same robot model. For achieving this, they restructured their code architecture. A core part was developed, which can be used in both leagues, with the goal of having as much code as possible in this core part. For the league specific requirements and parametrizations, another code part was built, which was used beside the core. Because of the restructuring, it was then possible to compare the robot's behavior in simulation and reality. So it was observed that the actuators and sensors, as well as the physics acting on the robot, differ profoundly. Through analyzing these differences, the simulator can be optimized to close the Reality Gap.

The RoboCup standard platform league team B-Human pursued an approach to achieve a more realistic simulation [22]. They tried to learn the parameters needed to more accurately simulate the actuators and their interaction with the environment by using genetic algorithms. The goal of this was to get an as realistic as possible simulation model. The learning has been done in different steps. First, the PID values and the maximum velocity were learned. A second step was performed to find the torque values of the motors. The final step was conducted to optimize the already found values and to obtain the parameters needed by the physics engine ODE to simulate friction and

contacts realistically. For the learning, the robot had to perform different motions, which were compared to the same motion execution on the real robot. In contrast to other approaches, in this one, the PID values were learned to match the real robot's behavior more accurately instead of using the same PID values in simulation and reality.

The RoboCup humanoid league team MRL-HSL modeled and simulated their robot based on Simulink [O36] to achieve a realistic motion behavior of the robot in simulation [15]. The main work was done by investigating the correct simulation parameters to model their robot. Due to this work, a simulation of their robot was achieved, which is similar to the real robot. Nevertheless, it is not mimicking the real robot exactly.

Summary

The Reality Gap is a problem influencing lots of different areas in robotics. For several reasons, e.g., using AI methods or studying a robot's behavior, an accurate dynamic simulation is needed. Several approaches to minimize or bridge the existing difference between simulation and reality have been presented. For most researchers, the first step to an accurate dynamic robot simulation is to acquire the correct simulation parameters for the used robot. After this is done the variety of what still can be improved is enormous, from improving the physics engine to creating a more realistic simulation with the usage of AI techniques. The validation of the presented approaches is mostly conducted by comparing the joint angles and sometimes joint velocities and torques of an in simulation and reality executed motion.

3. Fundamentals

In this chapter, an overview of the fundamental aspects of this work, as well as the used software and hardware, is given. At first, the hardware (Section 3.1) is explained by presenting the Wolfgang robot platform and taking a closer look at the, in this thesis, relevant components of the robot. The force/torque sensor used for measurements is also described. The Wolfgang robot platform is currently used by the RoboCup teams Hamburg Bit-Bots and WF Wolves for competing in the RoboCup soccer leagues humanoid kid size and humanoid teen size, where the robot is playing soccer autonomously. The software of the Wolfgang robot platform is based on the ROS framework. Therefore ROS is explained in Section 3.2. Within this framework, lots of software has been developed by the Hamburg Bit-Bots to control their robot. The for this thesis relevant parts are also introduced in Section 3.2, as well as the other used software. In the next Section (3.3) the physical properties of the robot's links are described. The meaning of the properties, relevant for this thesis, is explained, and it is shown how they can be calculated. The last Section (3.4) deals with the functionality and the physical characteristics of DC motors as they are used for the actuation of the Wolfgang robot platform and have been studied as part of this thesis.

3.1. Hardware

This section begins with the introduction of the Wolfgang robot platform [O38]. It is described how the robot is constructed and what its general features are. After that, a closer look is taken on the motors integrated into the robot. For this, it is explained how the motor is constructed, and the motor's specifications are stated. During the thesis, the motors were studied further with the help of a force/torque sensor, which is introduced at the end of the section.

3.1.1. Wolfgang Robot Platform

The Wolfgang robot platform is depict in Figure 3.1. It is a humanoid robot, which was developed by the RoboCup humanoid league team WF Wolves based on the NimbRo-OP robot platform [35]. Some further adaptions were made to this robot platform by the Hamburg Bit-Bots. The primary computing unit is an Intel NUC. For the image processing on the robot, an Nvidia Jetson TX2 is integrated. Furthermore, an Odroid

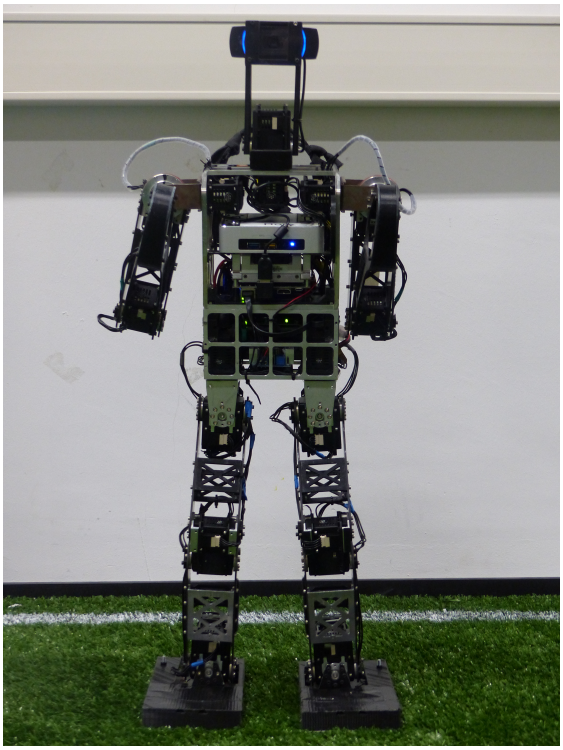


Figure 3.1.: A front view of the Wolfgang robot platform is shown.

XU4 is incorporated as an additional computing unit. Few sensors are integrated into the robot as these are limited to sensors mimicking humanoid senses to be allowed to participate in the RoboCup humanoid league. A camera, an IMU and foot pressure sensors are integrated as external sensors in the robot. The robot uses an accumulator as the supply of power during a soccer game.

The robot is composed of 21 links made out of aluminum, carbon, and polylactic acid (PLA). The links are connected with joints consisting of the electrical actuators. Therefore the robot has 20 degrees of freedom (DoF), which are divided into six DoF in each leg, three DoF in each arm and two DoF in the head. This division represents the five kinematic chains of the robot, which are connected through the robot's torso. This is illustrated in Figure 3.2. The actuators integrated into the legs are Dynamixel MX 106 motor because they can produce higher torques than the Dynamixel MX 64 motors, which are mounted in the arms and the head. The motors are connected over an RS-485 bus and are parallel-connected with cables. The communication between the computing unit and the motors themselves is realized with a Rhoban DXL Board [O4], which is a communication board designed for humanoid robots using Dynamixel motors by the RoboCup team Rhoban Football Club.

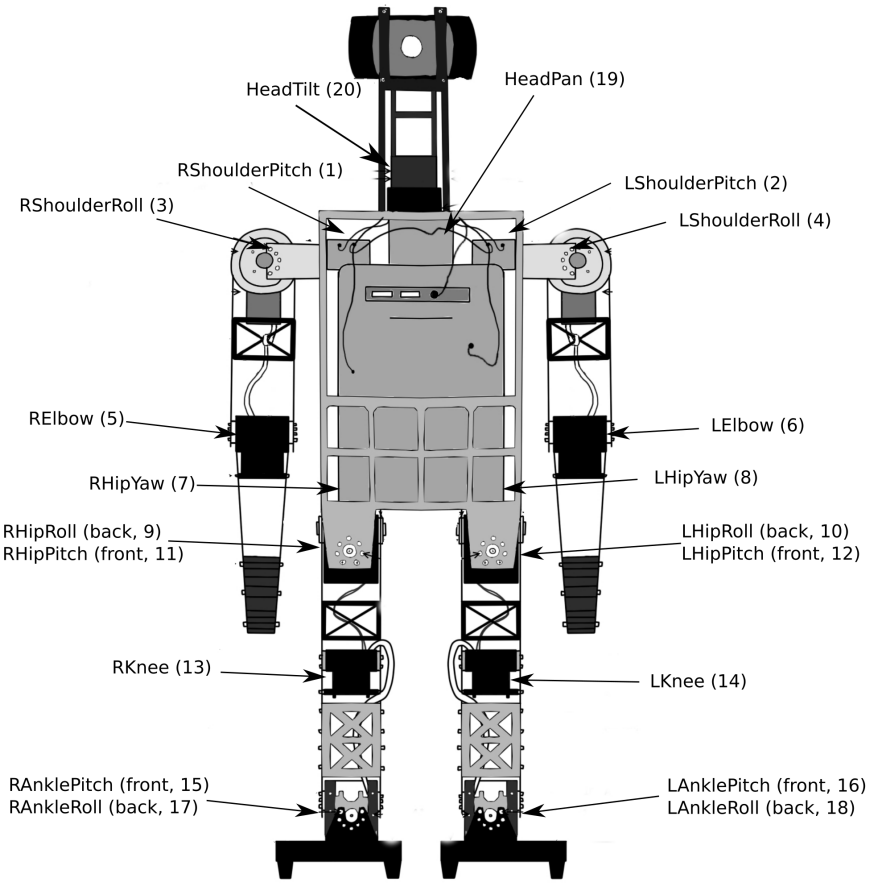


Figure 3.2.: This figure shows the kinematic structure of the Wolfgang robot. All joints integrated into the robot are labeled with their names and number. The figure is a modified version of [O10].

3.1.2. Dynamixel Motors

The motors integrated into the robot are Dynamixel MX 64 [O17] and MX 106 [O16] motors manufactured by Robotis [O28]. An MX 106 motor is shown in Figure 3.3. Both motor types are servo motors, which are composed of a brushed DC motor, a gearbox, a position encoder, and several other electronic components for measurements, control, and communication over the bus system. Therefore both motors work with the same mechanism. The motors distinguish in some specifications, which are shown in Table 3.1. The main difference is that the MX 106 motors can produce higher torques due to a higher gear ratio and a stronger DC motor. This leads to the MX 106 motor being slightly bigger. The gear ratio of the gearbox influences the transmission of the by the DC motor provided input torque and velocity to the output torque and velocity at the motor horn.

Figure 3.3.: A Dynamixel MX 106 motor is shown. At the front, the motor horn is on view. On each side, a plug for the cable is located. A bearing is located at the back.

	MX 64	MX 106
Backlash	$0.33°$	$0.33°$
Gear Ratio	200 : 1	225 : 1
Stall Torque (14.8V)	7.3 Nm	10.0 Nm
No Load Speed (14.8V)	78 rev/min	55 rev/min

Table 3.1.: A comparison of the for this thesis relevant specifications, as the data-sheet claims, of the two servo motors integrated into the Wolfgang robot, is shown. On the left, the values of the Dynamixel MX 64 [O17] are presented, while the ones from the Dynamixel MX 106 [O16] are presented on the right.

For example, the gear ratio 200:1 of the MX 64 means, that the DC motor turns 200 times faster than the motor horn, which in return can apply more torque. Due to the usage of gears, it is not possible for the motor to exactly move to the defined position because of the backlash. That is an effect which occurs because of imprecise meshing of the gears [O21]. Furthermore, the usage of a gearbox can increase friction, inertia, and torque ripple [25]. The DC motor and its influences are further explained in Section 3.4. The servo motors are internally controlled by a PID controller. For the motor control the position control mode is used by the Hamburg Bit-Bots. Other operation modes, e.g., the velocity control mode are also available. The deployed communication protocol is the Dynamixel Protocol 2.0 [O6].

3.1.3. Force Torque Sensor

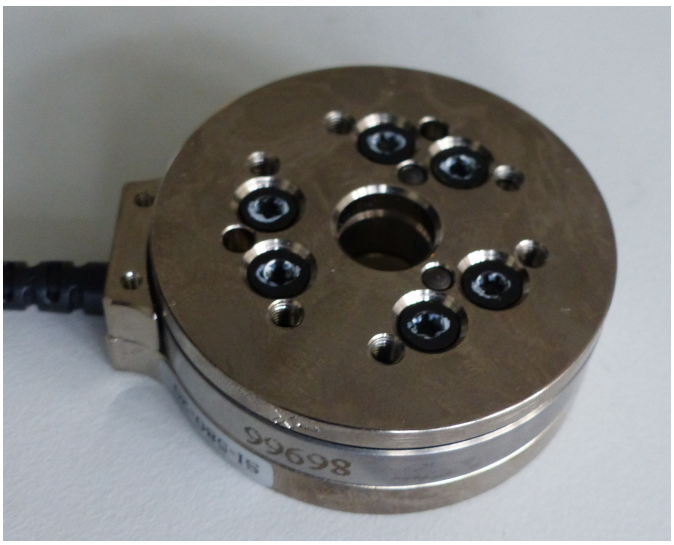


Figure 3.4.: The used multi-axis ATI force/torque sensor Mini 45 [O37] is shown. On the top, a tool can be mounted and on the bottom is the mounting side. In the middle, the sensor itself is located, connected to a Netbox with the cable on the left.

In this thesis, the ATI force/torque sensor Mini 45 [O37] was used, which is shown in Figure 3.4. It is a multi-axis sensor, which measures force and torque in the three dimensions. As transducer multiple silicon strain gauges are used to detect forces in the six axes. Biasing is a convenient way to exclude the sensor bias and the tool weight from the measurement results by introducing an offset to measured values. Software Tools for the biasing are provided. Furthermore, this sensor is ROS integrated with the help of an interface provided in the ROS package *netft_utils* [O29].

The sensor uses transducers to convert energy for transforming a signal from one type to another, here to an electrical signal. As mentioned the transducers in the sensor are strain gauges. A strain gauge consists of an elastic wire attached to the substrate [16]. If a force is applied to the substrate, this leads to an elongation of the wire because of the strain from the substrate. As a consequence, the resistance of the wire ascents because its cross-sectional area decreases due to the elongation. The difference in the elongation is small to protect the wire, which is why the resistance change is also small. Hence a Wheatstone-Bridge [11] is used to track the resistance. As the strain is proportional to the applied force in the usage range of the sensor, the resistance change is a measure for the applied force.

3.2. Software

This section introduces the software used in this thesis. At first, ROS is presented as the framework influences further software. After that, the simulator Gazebo is presented. For the simulation purpose, it is necessary to model the robot through a robot description file. Therefore the in the ROS environment commonly used description file syntax is introduced afterward. Next, the CAD Design Program Autodesk Inventor is presented, which was used to improve the above mentioned description file. Finally, the relevant parts of the codebase from the Hamburg Bit-Bots are described.

3.2.1. Robot Operating System (ROS)

ROS [O27] is a framework for robot programming. It was designed to realize the following goals: handling multiprocessing, handling multiple hosts, supporting various programming languages, providing lots of different tools, having modularity as well as being free and open-source [34]. Using ROS has the advantage of creating a hardware abstraction, providing low-level device control, and that the package management is dealt with, as well as the message-passing between processes [O14]. The user also benefits from a huge community providing software packages in various robotics tasks, which can easily be integrated into their software because of the hardware abstraction.

The core of ROS is the ROS Master. It manages the communication between all software packages. Each package runs in a so-called node. The nodes are independent of each other and can be executed parallel on one or multiple machines. They communicate with defined messages over so-called topics. Every topic handles just one message type. The nodes can send messages to the topic, or can get messages from other nodes on this topic. Furthermore, ROS provides actions, which realize telling another node to start running a process and being able to get feedback or to cancel the started process during the execution of it. It is possible in ROS to start one node or to start multiple nodes with one command with the help of a launch file. Further, it is also possible to record the traffic on the topics with rosbags. This can be used to test with this data or to get an insight into what happened on the robot by looking at the recorded data. Other concepts are supported by ROS as well but are not mentioned because they are not relevant for this thesis.

3.2.2. Gazebo

Gazebo [O34] is a robot simulation software, which allows simulating multiple robots and a matching environment within one three dimensional scenario [20]. It is an open-source project mainly developed by the Open Source Robotics Foundation [O23]. Different open-source physics engines are supported for the simulation of dynamics. By default ODE [O22] is used as a physics engine, but the usage of Bullet [O3], Simbody

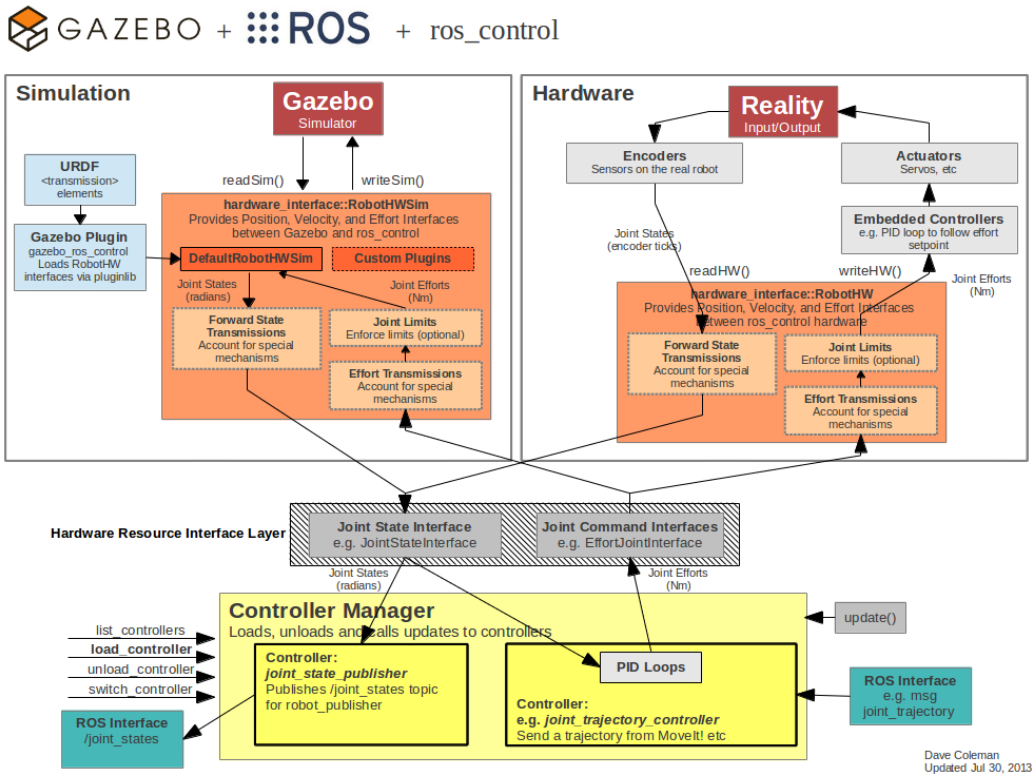


Figure 3.5.: Actuator Control through *ros_control* in simulation and reality is shown. It illustrates clearly that the same control loop is used for movement execution in reality and simulation except for the actual motor control and the needed Plugin for the usage of the software in Gazebo. The figure is a taken from [O24].

[O32] and DART [O5] is also possible. The worlds and models of robots and objects have to be described in the Simulation Description Format (SDF) [O33].

ROS is integrated into Gazebo through ROS packages which provide a wrapper for Gazebo itself by specifying the necessary interfaces to use the features of ROS directly in Gazebo [O30]. So it is possible to launch Gazebo like every other node in the ROS environment through a launch file. Moreover, it is possible to use the URDF (see Section 3.2.3) as the specification for the robot model description instead of creating an SDF just for the simulation purpose. This can be done with small adaptions to the URDF [33], which do not interfere with the usage of the URDF on the real robot. Furthermore, *ros_control* (see Section 3.2.5) can easily be used in the simulation in the same way as it is used in the real robot for controlling the actuators. This is shown in Figure 3.5.

Because of the ROS integration, it is possible to use the software used to control the robot in reality, also in simulation. Therefore the robot's behavior in reality and simulation can easily be compared. A disadvantage of Gazebo is that it does not include an accurate simulation of the actuator dynamics. An actuator model should, for example, include modeling motor characteristics like rotor inertia or gear ratio [40]. Still, Gazebo was chosen as a simulation environment because it has this ROS interface, and the simulation software is open-source. Furthermore, this simulator is used by other RoboCup teams as well. For example, a setup in Gazebo for the RoboCup Middle Size League was created [45], and for the 3D simulation league, a Gazebo plugin also exists [O15].

3.2.3. Unified Robot Description Format (URDF)

The URDF [4] defines how a robot model has to be described for usage within ROS. URDF is based on the XML format. It is usable for robots with rigid links, which are connected through joints in a tree structure. In general, the URDF consists of two main parts. The first part contains the description of the robot's links. Each link has to have a name. Additionally, there is the possibility to specify the inertial model for the physical parameters, the visual model for proper visualization of the robot, and a collision model for the collision characteristics of the link. Secondly, the joints have to be defined with a name and type, as well as the parent and child link the joint is connecting. Additionally, for joints of the type revolute or prismatic a `limits` tag, which has to contain a maximum effort and a maximum velocity for the joints movement has to be defined. Moreover, there exist optional parameters like the dynamics parameters where physical properties can be described. Besides the essential `link` tags and `joint` tags, it is possible to define further tags to extend the informative content of the URDF or gain the possibility of using the URDF in the simulator Gazebo (see Section 3.2.2).

As the focus of this thesis is on the dynamic parameters of the simulation model, just these parameters are described further. For simulation, the links have to have an `inertial` tag, as shown in Figure 3.6, because Gazebo requires it. It involves an `origin` tag, a tag for the `mass` value and one for `inertia`. The `origin` tag states the position of the center of mass concerning the origin of the link itself. The `mass` value defines the mass of the link. `Inertia` describes the inertia tensor relative to the center of mass. The characteristics of a center of mass and inertia tensor are further explained in Section 3.3. The description of the joints, shown in Figure 3.7, can include tags for physical constraints of them. The `dynamics` tag can be used to specify the damping and friction of the joint. The `limits` tag, which is described above, can also be used for joints of the other types. With the not required parameters lower and upper, describing the minimal respectively maximal position, which can be accessed, is also possible.

Useful extensions of the URDF for the simulation are the `gazebo` tag and the `transmission` tag. With the `gazebo` tag, it is possible to specify the information needed for

```
<link name="r_lower_arm">
        <inertial>
                <origin xyz="0.0018497 -0.00091671 -0.059095"
                        rpy="0 0 0" />
                <mass value="0.23502" />
                <inertia ixx="0.0011613" ixy="-5.8221E-07"
                         ixz="-7.4224E-05" iyy="0.0011747"
                         iyz="2.0082E-05" izz="0.00010273" />
        </inertial>
</link>
```

Figure 3.6.: An example of a link description in a URDF is shown containing the relevant part for this thesis. The example is composed of the relevant link name and the `inertial` tag for the specification of the links physical parameters.

```
<joint name="RElbow" type="revolute">
        <parent link="r_upper_arm" />
        <child link="r_lower_arm" />
        <limit effort="2.5" velocity="5.6548668" lower="-1.635"
               upper="1.5707963267910001" />
        <dynamics damping="1.0" friction="1.0" />
</joint>
```

Figure 3.7.: An example of a joint description in a URDF is shown containing the relevant part for this thesis. The example is composed of the mandatory joint name, `parent` link, `child` link, and the effort and velocity `limit` as well as the lower and the upper limit and the `dynamics` parameters.

an accurate simulation like adding sensors or cameras. Furthermore, the dynamics of links and joints can be described more precisely. This can be done by specifying further physical parameters which are used by the physics engine. The possible parameters are defined for SDF and can be set in an URDF in the `gazebo` tag. The `transmission` tag can specify how the actuator in a joint is controlled, and the mechanical reduction of a motor can be defined.

3.2.4. Autodesk Inventor

Autodesk Inventor [O1] is a 3D CAD software, which can be used for CAD Design and analysis of CAD designed parts. It is possible to assemble several designed components into one file and then analyze the whole construction. In this thesis, it was used to create an experimental setup for the analysis of the motors (see Section 4.3) and to

analyze the physical properties of the modeled robot's links.

The program can calculate the center of gravity and the inertia tensor relative to the center of gravity for every part saved in a file format from Autodesk Inventor. Because the used robot was already created and assembled in this program, it was not considered to use another CAD Design program. As a commercial software was used, the source code is not available. Therefore it is just known that the implementation of the calculation follows the formula developed from theory (see Section 3.3), while the exact implementation of the formula including the integration procedure is unknown [O7].

The parts and assemblies are modeled in Autodesk Inventor regarding the coordinate system they were designed in. Therefore to transfer the calculated physical properties to the URDF, the output has to be transformed into the robot's coordinate system in the ROS environment.

3.2.5. Hamburg Bit-Bots Codebase

The codebase of the Hamburg Bit-Bots is open-source and is published on Github [O11]. Out of all the packages implemented by the Hamburg Bit-Bots for their soccer-playing robots, only some have been used in this thesis. Therefore just these relevant packages are explained here. One of the used packages is the one for playing keyframe animations named *bitbots_animation_server*. For proper usage of this package, additionally the *wolfgang_animations* package is needed, which contains keyframe animations working on the Wolfgang robot platform. Furthermore, the packages for motor control were used.

The *bitbots_animation_server* parses json files containing keyframe animations and interpolates them, so the motor control software can execute the motion. Thereby one keyframe of an animation is played after the previous one is finished. The package provides a ROS action to play an animation. Examples for animations which can be parsed by the software package can be found in the *wolfgang_animations* package. It contains the keyframe animations of the Wolfgang robot platform which are used for playing soccer and some further animations. In general, the structure of an animation file is that it lists one keyframe after the other. For each frame, the position of all motors has to be defined, as well as the duration of the frame and the pause after the frame was executed.

The motor control is realized by several packages. In the ROS environment the *ros_control* package [8] is used for controlling the motors and sensors. Furthermore, the package contains controllers which represent the interface between software and hardware. The Dynamixel packages are responsible for translating the commands from *ros_control* to the motor protocol. Overall it is possible to read information from the motors regarding, for example, their position, velocity, torque, temperature, and input voltage. As the motors are used in position control mode, a new position for a motor stated by *ros_control* will be written to the motor through the mentioned lower-level

software and hardware. The Hamburg Bit-Bots use these packages combined with a self-implemented hardware interface and controller.

3.3. Physical Properties of Robot's Links

Besides stating the mass, the rigid links of the robots have to be physically characterized through its center of mass and inertia tensor. The center of mass is the point of a link where an applied force creates a linear acceleration and not also a rotational one, and it is the point on which the link is balanced [19]. The calculation of the center of mass is in general done with the following formula:

$$\mathbf{R} = \int_V \mathbf{r} \frac{1}{M} \rho(\mathbf{r}) dV \tag{3.1}$$

The in the formula used parameters are representing:

R coordinate vector of the center of mass

r position vector

M total mass of the object, in this case, the link's mass

ρ density

dV volume element

Due to the integration the weighted sum over the position vectors is calculated, which results in the center of mass. The position vectors are weighted with the amount of the total mass of the corresponding infinitesimal volume element.

The origin of the reference coordinate system of the inertia tensor is the center of mass [O20]. The inertia tensor describes the mass distribution in the link. It represents the analogon to mass in a linear movement. It is used to calculate the necessary torque for rotation by multiplying the inertia tensor with the angular velocity. In general, the inertia tensor is calculated with the following formula [O8]:

$$\mathbf{I} = \int_V \rho(\mathbf{r}) r_\perp^2 dV \tag{3.2}$$

The in the formula used parameters are representing:

I inertia tensor

r position vector

ρ density

$r_\perp$ perpendicular distance from the axis of rotation

dV volume element

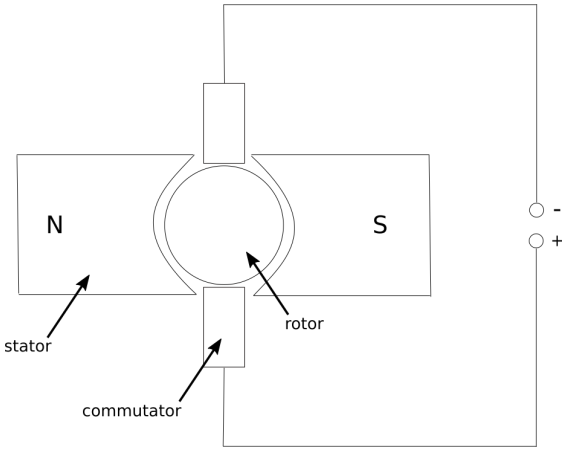


Figure 3.8.: A schematic structure of a DC motor is shown. The three main components, stator, rotor, and commutator, are labeled. This graphic is a modified version from a figure in [12].

3.4. Functionality of a DC Motor

A DC motor is the actuator of a servo motor by producing torque, which is passed to the motor horn of the servo motor through the gears. The functionality of the motor is explained in Section 3.4.1. As the DC motor has a strong influence on the actuation of the robot, the physical impacts of a DC motor are considered in Section 3.4.2.

3.4.1. Functional Principle

A DC motor turns electrical energy into rotational mechanic energy [12]. The motor consists of a magnetic stator, a rotating coil, and commutators. A schematic structure is shown in Figure 3.8. Current flows in the coil, which leads to a magnetization of the rotor. Because the rotor is located in a magnetic field, it starts to turn due to repulsion. For getting a continuous rotation, the commutator is needed to reverse the polarity of the coil [37]. Otherwise, the coil would perform a half rotation cycle because it then has reached its position of equilibrium. The coil turns a little further than the position of equilibrium because of inertia. If the polarity is then reversed, a torque is acting on the coil to send the coil into the new position of equilibrium, which is a half rotation cycle away. Because of Faraday's Law of induction, the rotation of the coils induces a voltage [2]. This voltage works against the applied potential due to Lenz's law. Therefore the armature current is affected. As a consequence, the power consumption depends on the mechanical load. If the motor has a higher load attached, it turns more slowly and therefore the from the amount of the from the power supply taken current increases. Usually, multiple coils are included in a DC motor to smooth the voltage.

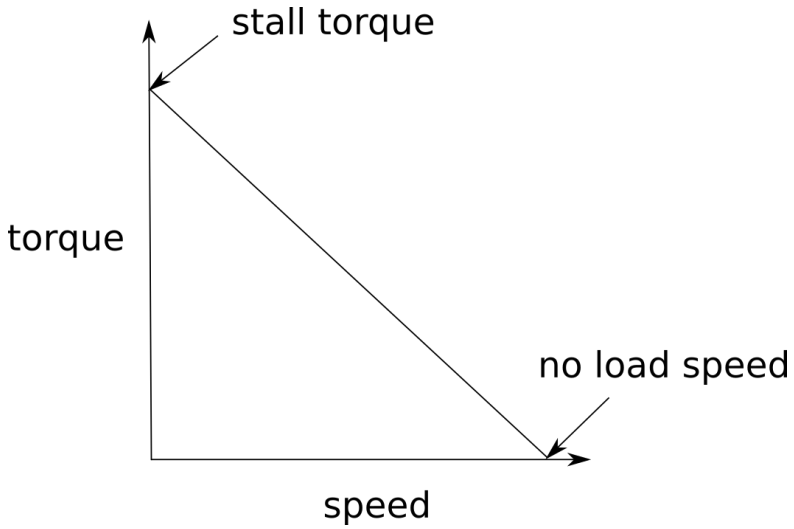


Figure 3.9.: This graph shows the connection between the torque and angular velocity of a DC motor. It is shown that the torque behaves negative linear to the speed. Furthermore, the characteristic stall torque and no-load speed are shown. This graphic was created according to the model shown in [O19].

3.4.2. Physical Influences

The torque and velocity produced by a motor are connected [O19]. The torque is negative linear to the rotational speed, which is shown in Figure 3.9. Stall torque is the highest torque a motor can produce and occurs when the motor is not turning anymore due to a high load. If no load is attached to the motor, the highest rotational speed occurs, which is named no-load speed. Therefore with attaching a load to a motor, it is possible to influence the torque and speed of a motor as load drains energy from the motor leading to a slower rotation. The maximal mechanical power is reached at half of the no-load speed [O19].

The number of revolutions of the motor has to produce a voltage balance between the applied and induced voltage in no-load running [37]. With that the following equation can be derived for the number of revolutions of the engine:

$$n = \frac{U - I \cdot R}{c \cdot \Phi} = \frac{U}{c \cdot \Phi} - \frac{I \cdot R}{c \cdot \Phi} \tag{3.3}$$

The in the formula used parameters are representing:

n number of revolutions of the motor

U voltage

I armature current

R armature resistance

c machine constant

Φ total flux of the electric circuit

As during no-load running the armature current is zero, it can be inferred that the number of revolutions is about $\Delta n = \frac{I \cdot R}{c \cdot \Phi}$ smaller than the no-load speed when the load or current is increased. This shows that the speed is linear decreasing with increasing current. Furthermore, it can be followed that the no-load speed is higher when the input voltage is higher.

The torque is dependent on the armature current, the total flux, and a machine constant [37]:

$$M = I \cdot \Phi \cdot c \tag{3.4}$$

The in the formula used parameters are representing:

M torque

I current

Φ total flux of the electric circuit

c machine constant

As the DC motor is separately excited, the total flux is unaffected by load changes. Therefore is the torque proportional to the armature current, while as shown above the speed is negative linear to the armature current. The current depends on the voltage provided by the power supply. This leads to the fact that a higher torque can be produced when the input voltage is higher.

The influence of input voltage was also mentioned by Kubisch [21]. It was observed that the voltage supply decreases under a load of operating an actuator with high torque slightly. When a humanoid robot is getting up from the ground this effect adds up from several motors using high torques because they all use the same power supply, which resulted in an about 25% decreased voltage in the study. This can lead to the effect that the motor's torques are not high enough anymore for the robot to get up.

Furthermore, the inertia effects of the rotor or friction during the rotation are influencing the performance of the motor. Another option is to change the windings of the coil, which increases the repulsion. Moreover, due to the power conversion, torque fluctuations can happen [25].

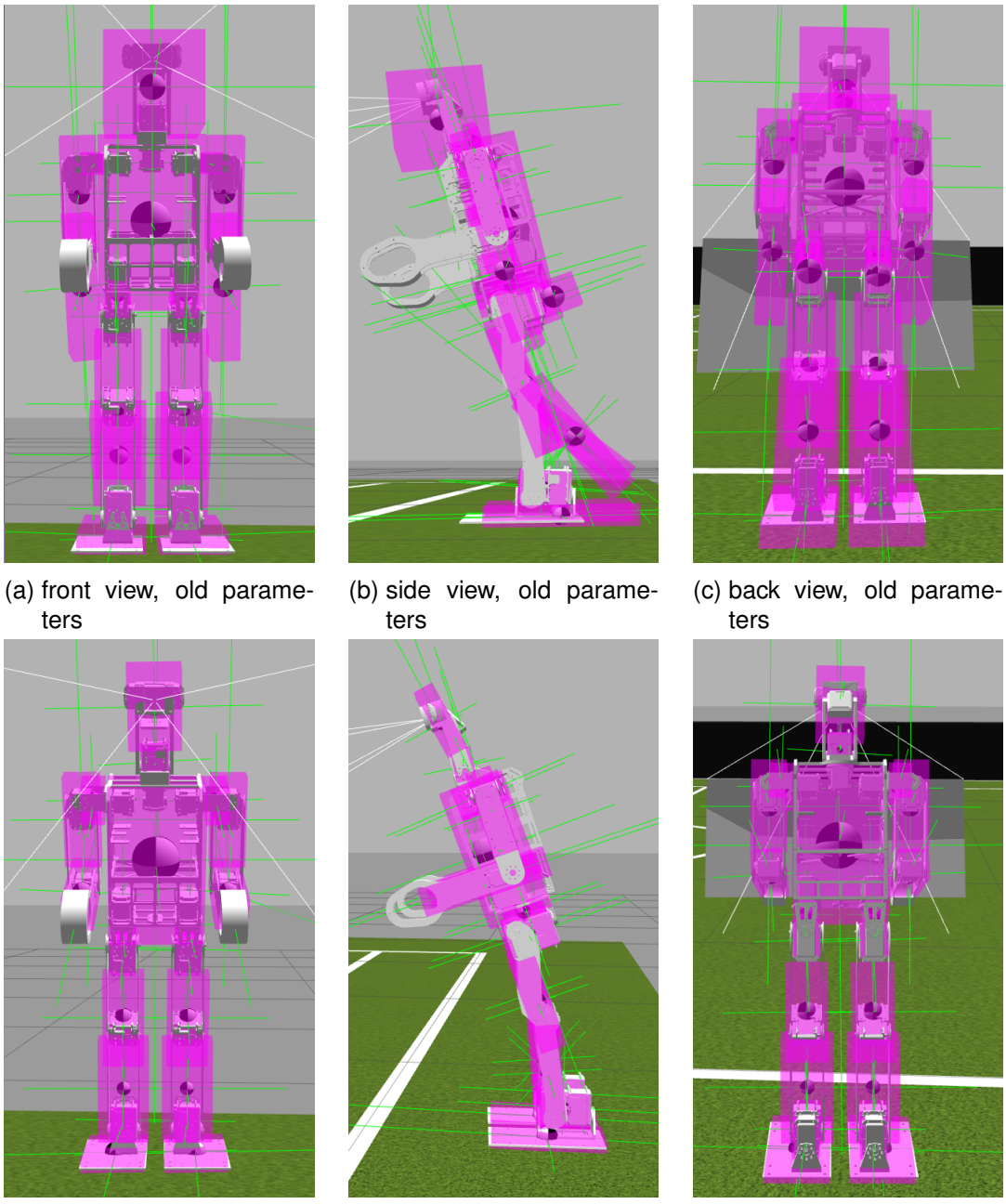
4. Improvements for the Dynamic Simulation Model

In Chapter 2 the inference was drawn, that the first step to close the Reality Gap should be the improvement of the simulation parameters of the robot. Therefore the improvement of the dynamic simulation model of the Wolfgang robot platform is presented in this chapter. In Section 4.1 a first step for minimizing the Reality Gap is taken with the enhancements of the physical characterization of the robot's links in the robot description file. Besides the links, the joints take a significant part in the robot's model. The joints consist of motors, thus it is necessary to analysis the actuating motors. To do so, the input voltage of them had to be examined as the power supply influences the actuator, which is explained in Section 3.4.2. So in Section 4.2, the input voltage of the in the robot integrated motors is studied under different load conditions. After that, the motors itself are investigated with the previous results. Thus the analysis of a motor's behavior under different load conditions is described in Section 4.3. At the end, in Section 4.4, the improvements made are summarized.

4.1. Inertial Link Parameters

In this section, the revision of the inertial parameters, mass, origin, and inertia, which are stated in the URDF (see Section 3.2.3), is described. For the Wolfgang robot platform, a URDF already existed, which had updated visual and collision parameters for the links. The status of the physical inertial parameters was unknown. Therefore the quality of the given parameters was evaluated.

In Figure 4.1 on the top the inertial parameters origin and inertia of the links from the previous URDF are visualized, which were inaccurate. For example, this is shown in Figure 4.1a where the inertia tensor of the head link, whose position can be abstracted from Figure 4.2, is too large. Furthermore, the center of mass of the head is located too far down, as the heavy part of the head is the camera. Another confirmation is shown in Figure 4.1b where the center of mass and the inertia tensor of the lower leg link are wrongly located in the coordinate system because of an orientation change of the visual. Furthermore, in Figure 4.1c, it is shown that center of mass and the inertia tensor of the lower arm link do not fit the link itself as they are located in another direction in comparison with the link.



(a) front view, old parameters
(b) side view, old parameters
(c) back view, old parameters
(d) front view, new parameters
(e) side view, new parameters
(f) back view, new parameters

Figure 4.1.: A visualization of the center of mass (black and white sphere) and the inertia tensor (purple box) of the links from the Wolfgang robot is shown. In the top row, the values stated in the old URDF are visualized and in the bottom row, the ones stated in the URDF after the adaption of the inertia link parameters.

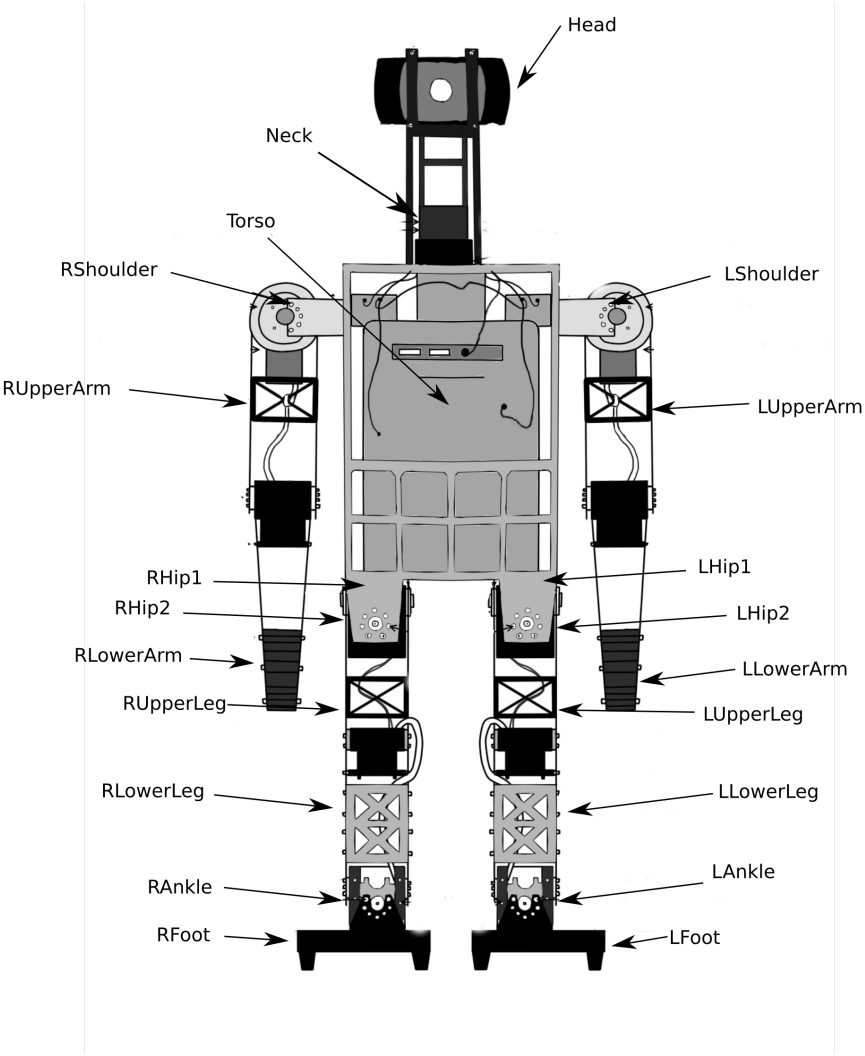


Figure 4.2.: This figure shows the link structure of the Wolfgang robot. All links integrated into the robot are labeled with their names. The figure is a modified version of [O10].

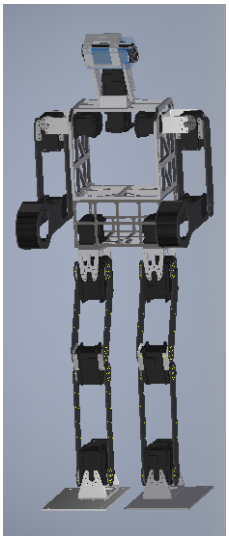


Figure 4.3.: This figure shows the already existing robot model in Autodesk Inventor.

As a consequence of these results, the origin and inertia parameters of the URDF had to be reevaluated. For this, the inertia tensor and center of mass was determined with the help of Autodesk Inventor (see Section 3.2.4). To be able to confirm these results, the center of mass was measured in an experimental setup by hand as well. As the structure of the Wolfgang robot is quite complicated, the Autodesk Inventor results for the inertia tensor were not reviewed. For a complete check of the inertial parameters, the mass of the links was measured after the update of the two other parameters.

The determination of the inertia tensor and the center of mass by use of Autodesk Inventor needs the links designed in CAD format. In general, this already existed for the Wolfgang robot, but there were some components missing which are relevant for identifying the physical parameters. A picture of the existing model is presented in Figure 4.3, revealing the missing parts in the robot model when this is compared to the picture of the real robot in Figure 3.1. For example, the computation units in the torso, as well as the leg spacers, are missing. The models of the links were adjusted to get a more accurate model by adding the missing parts as well as screws and nuts. For all parts of the robot, the density value had to be assigned to achieve the correct calculations in Autodesk Inventor. Therefore the density for the different materials integrated into the Wolfgang robot had to be determined. One part from each material was weighed, and the density was calculated. After that, the materials just had to be assigned to the robots parts in Autodesk Inventor, and the inertia tensor and the center of mass could be calculated.

To be able to evaluate the obtained values, the center of mass was measured in reality. The measurement was done with a compensator, illustrated in Figure 4.4. It was designed and 3D-printed for the thesis. Due to the symmetric structure of the Wolfgang

	average difference [m]
x direction	0.005
y direction	0.002
z direction	0.007

Table 4.1.: The average difference of the position of the center of mass between the value calculated by Autodesk Inventor and the one measured by hand is shown for the three dimensions.

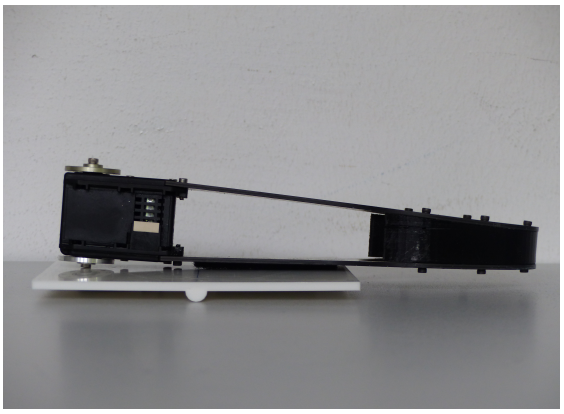


Figure 4.4.: This picture shows the experimental setup for the center of mass measurements in reality, at the example of the lower arm. Each link was balanced out on the compensator in three dimensions.

robot, just one arm and leg had to be considered besides the head and torso. The results of these measurements can be found in Appendix A.1. The average differences of the position of the center of mass between the measured values and the ones calculated from Autodesk Inventor are presented in Table 4.1. As the average differences are small and manual measuring is slightly imprecise, these measurements confirm the usability of the values from the program.

As it is shown now that the by Autodesk Inventor calculated values are accurate, they just had to be added to the URDF. Thereby it is necessary to regard the rotations of links when the visual is rotated. So in case of a rotation, the new values had to be rotated with the same angle. The result of the calculation can be seen in Figure 4.1 in the bottom row. In the visualization, it is visible that the center of mass of each link is now located at a reasonable place, and that the inertia tensor fits the links relative to the related center of mass as well. That the hands of the robot are not integrated into the visualized inertia tensor of the lower arm link originates from the structure of the link. The part with the highest mass and therefore the one most affecting the inertia modeling is located in the elbow in the form of the motor. The in comparison, to the motor, small weight of the hand does not have a high effect and is therefore not integrated in the visualization. This is also visible at the upper arm link, head link, and the upper leg link,

	MX 64	MX 106
maximum input voltage	16.0 V	15.8 V
minimum input voltage	12.8 V	12.7 V

Table 4.2.: The table shows the occurred minimal and maximal input voltages at the servo motors MX 64 and MX 106, which are integrated into the Wolfgang robot platform. The observation was done while the robot was connected to an accumulator, which is also used in games as a power supply. The results consist of values the motors gave as output about their input voltage.

.

where the mass distribution is also not uniformly distributed due to their structure. The other links of the robot are all mostly covered by the inertia tensor visualization because their mass distribution is almost uniformly distributed.

For a complete review of the `inertial` tag in the URDF just the mass values still have to be checked. The weighted values and the one stated in the given URDF can be found in Appendix A.2, as well as the relative error from the new values compared to the stated ones. The table shows that out of twelve different links, three of them were stated almost accurate with a difference of less than three percent. It is also shown that the shoulder link and the foot link were given wrong mass values as the relative error is over a hundred percent. Therefore also checking the mass values was important as the `inertial` tag is now a lot more accurate than it was before.

4.2. Input Voltage Study

This section deals with the analysis of the input voltage of the servo motors integrated into the Wolfgang robot platform. As explained in Section 3.4.2, the input voltage influences the maximal torque and velocity, which can be produced by the motor. As the behavior of this motor is later on analyzed, it is necessary to study which input voltages are applied to the motors to execute meaningful experiments with them. Therefore the maximal and minimal input voltage occurring at the two motor types, when the robot is using an accumulator as power supply, was observed over the discharging of an accumulator. For this, the output from the motors, what they state as their input voltage, was taken into account. The results can be found in Table 4.2. It is noticeable that the two maximum values are slightly different, although the motors are all connected in parallel. It is known that the voltages drop a little to the end of the longer chains. Still, interestingly, there is already a difference between the first motors in the leg chains and the ones in the arms and head chain. Therefore it was decided to study the input voltage of the motors further. In the following, it is described how the accuracy of the output values from the motors under different load conditions was examined.

For checking the output values from the in the robot integrated motors, the voltage of

	mean difference	maximum difference
low load	0.14 V	0.31 V
medium load	0.09 V	0.24 V
high load	0.24 V	0.6V

Table 4.3.: The table shows the results from the analysis of the accuracy of the from the motors measured input voltage values in comparison to the hand-measured voltages parallel to the motors. The mean and maximum difference of the values is shown for the three load conditions the experiment was performed with. The low load condition is realized by a hanging robot, in the medium load condition the robot is standing, and in the high load one, it is walking. The values of the walking experiment are conducted with fewer data points.

the motors was measured by hand. This was done with a voltmeter which was parallel-connected to the regarded motor. Then the measured voltages were compared with the output from the motors. As the available voltage depends on the load of the robot, the measurements were done for three load conditions. The first represents a low load, as the robot was hanging. Secondly, the robot was standing as it is shown in Figure 3.1. For these two conditions, the voltage was measured for each motor. Finally, the input voltage was evaluated under high load, while the robot was walking. Thereby the voltage was just measured at the end of the five kinematic chains, because it is technically hard to measure in between the chains, while the robot is walking and the added value is not that big as the voltage drop at the end of the chains is the most interesting.

Overall the experiments showed that the leg chains experience a higher voltage drop than the other chains. Furthermore, it was confirmed that under high load, the voltage decreases. Table 4.3 compares the values measured by hand, and the output values from the motors. It shows that the motors give out the most accurate values under medium load. Still, there is a significant difference between the measured data and some motors output of the input voltage. The symmetrically arranged motors can also give out significantly different data while being in a similar situation. For instance, during the measurement under medium load condition, the LElbow joint stated a 0.24V too high value, while the RElbow joint was stating a value 0.08V too low. Under small load condition, both values were wrong about the same amount of 0.2V.

Each motor has its measurement error sources. Therefore it is not merely possible to state the error relationship of the values of the motor to the real values. Still, the values outputted from the motors are overall usable to get an overview of the applied voltage to the robot. Due to the not completely precise values, the maximum and minimum voltage under accumulator usage from the MX 64 motors will be used for both motors as the difference is small and through using the same voltage values, a better comparison of the two motors is possible.

4.3. Physically Influenced Joint Parameters

For improving the joint parameters of the URDF, which depend on physics, these values first had to be measured at the motors. Therefore it is described in this section how the torque, velocity, and static friction parameters were measured. The experiments were done under different applied voltages. The minimum and maximum measured voltage in the robot under accumulator usage was used, as described in Section 4.2. Furthermore to be able to compare the values with the from Dynamixel provided datasheet also 14.8V was used as voltage. The analysis was performed for the MX 64 and MX 106 motors. It was planned to use three motors each, whereby two of them were already used, and one was completely new. However, the results from the new MX 106 were different from the old ones, while for the new MX 64 motor this could not be observed. Therefore a fourth MX 106 motor, which is also new, was additionally tested to confirm the results of the third motor. With that, it can be evaluated if a difference is present between the different aged motors.

The experimental setup is shown in Figure 4.5. The motor is clamped to the table. Attached to the motor is a lever for hanging different load onto it. At the end of the lever, an IMU is attached for measuring the angular velocity. The measurement is realized with the gyroscope included in the IMU. The angular velocity of the motor can be measured at the end of the lever because it is located on the same axes of rotation and the angular velocity is independent of the distance to the center of rotation. The other measured variable is the torque of the motor. This is realized with the force/torque sensor presented in Section 3.4, which is directly connected to the motor horn. For every motor, the occurring torques and velocities are measured for different loads until stall torque is reached, which is noticeable through overload errors of the motor. Overload Errors are thrown to protect the motor when the load is too high for the motor to move. This test execution is done with the three mentioned voltage values set at the voltage supply.

The identification of static friction is realized based on the experiment presented by Mensink [28]. The static friction can be determined by measuring the torque when the motor starts to turn due to a force acting on it. Different from the related experiment, the torque was directly measured with a force/torque sensor, and the force was applied with a load instead of a spring scale. As an experimental setup, the same one as in the previous experiments was used.

The experiments regarding the torque and speed of the motors confirm the connection between these two values stated in 3.4.2. The results are shown in Figure 4.6. For the no-load speed, the torque value is negative because the torque acts in the opposite direction when no load is attached to the lever. The analysis of the measured values for each motor type and each voltage value was done by applying linear regression. With that, it was possible to determine the stall torque and no-load speed. The values for the graphs are the maximum angular velocities and the corresponding torque values per load. Interestingly, the resulted stall torque is on average about 1 Nm smaller than

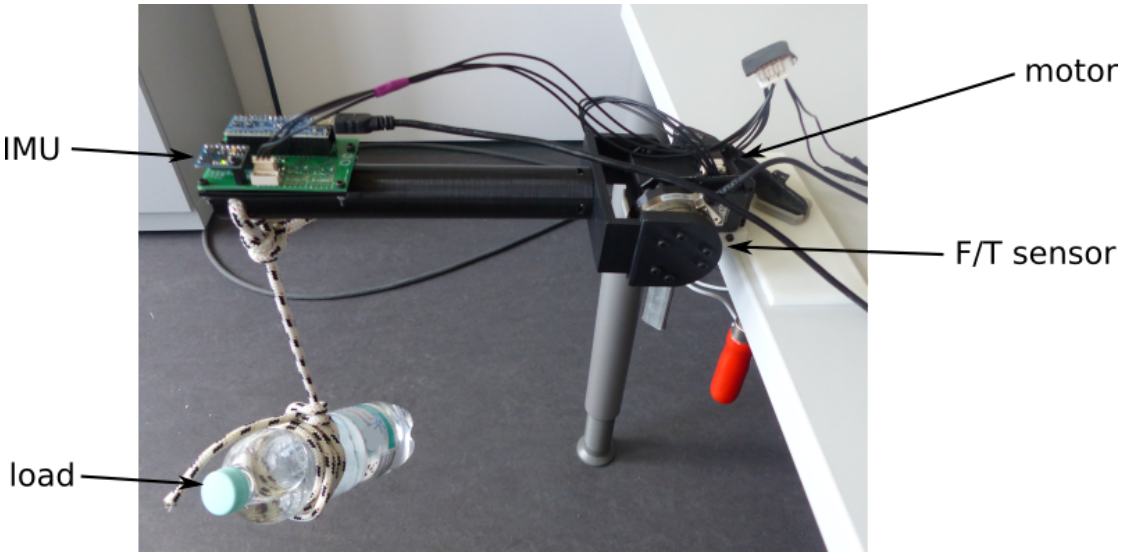
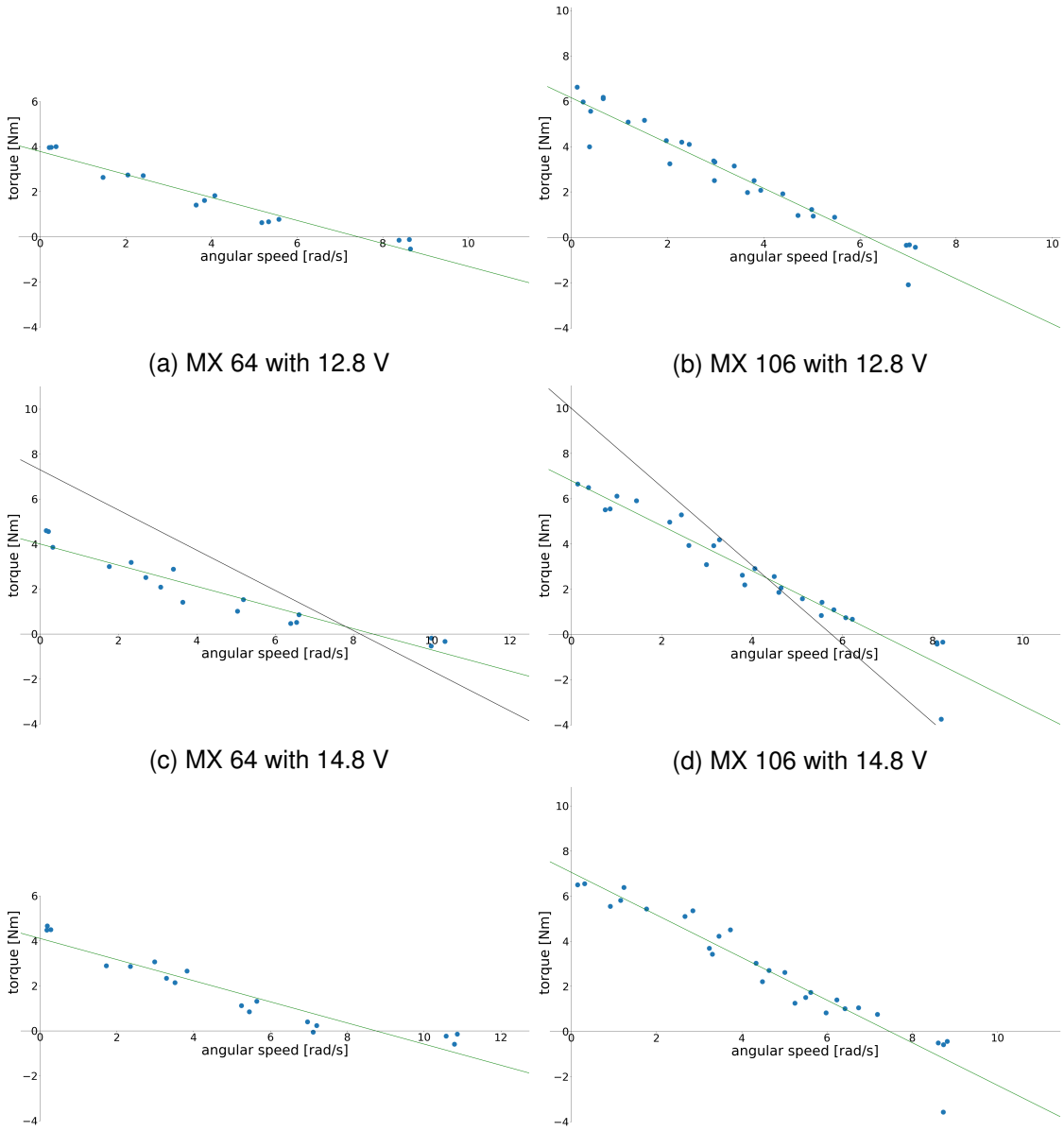


Figure 4.5.: This figure shows the experimental setup used for the analysis of the motors. The motor on the right side is clamped to the table. Attached to the motor horn on the front is a force/torque sensor. Connected to the sensor and the bearing is a lever about a bridge. At the top of the lever is an IMU on a motor control board located for velocity measurements. For applying different load to the motor, a load can be added to the front of the lever. The motor is connected to a voltage supply and controlled by the motor control board.

the maximum measured torque, which is shown in Table 4.4. Moreover, the maximum measured speed is about 1.5 rad/s higher than the determined no-load speed, which is shown in Table 4.5. In Figure 4.6c and 4.6d a comparison between the measured values and the manufactures data is shown. A significant difference between the values from the data-sheet and the measured ones is observable. The stall and maximum torques are also stated in Table 4.6, which also shows the deviation of the values. The no-load speed values are almost accurate, while the stall torque differs for the MX 64 about 45% and the MX 106 about 32%.

As a result of this experiments, the maximal occurred angular speed, and torque were written into the URDF because that is what is expected in the URDF. The other possibility would have been to take the by linear regression determined no-load speed and stall torque. However, the determined stall torque does not represent what maximum torque the motor can produce because higher torque values were measured. It seems that the linear relation between torque and angular velocity is only a good approximation. The values were taken from the 14.8V experiments as this present almost the middle of the battery operation range.

An interesting observation was made during the torque and speed experiments. While



(a) MX 64 with 12.8 V

(b) MX 106 with 12.8 V

(c) MX 64 with 14.8 V

(d) MX 106 with 14.8 V

(e) MX 64 with 16.0 V

(f) MX 106 with 16.0 V

Figure 4.6.: The results from the motor experiments regarding torque and speed are shown in the graphs. The speed in rad/s on the x-axis is plotted against the torque in Newton-meter on the y-axis. The blue dots show the maximum velocity and the corresponding torque value for a particular load attached to the motor. With the blue dots, a linear regression was performed, which resulted in the green line. For the measurement with 14.8 V, manufacturer's data is available for comparison, shown through the black line. Three MX 64 motors and four MX 106 motors were used to obtain these results.

		maximum torque [Nm]	stall torque [Nm]
MX 64	12.8 V	4.85	3.78
	14.8 V	5.43	4.00
	16.0 V	5.762	4.111
MX 106	12.8 V	7.06	6.15
	14.8 V	7.57	6.80
	16.0 V	7.88	7.06

Table 4.4.: The table shows the maximum occurred torque per motor type and voltage value for all used motors and the out of these values with linear regression determined stall torque.

		maximum speed [rad/s]	no-load speed [rad/s]
MX 64	12.8 V	8.64	7.42
	14.8 V	10.33	8.49
	16.0 V	10.86	8.74
MX 106	12.8 V	7.15	6.16
	14.8 V	8.22	6.82
	16.0 V	8.82	7.47

Table 4.5.: The table shows the maximum occurred angular speed per motor type, and voltage value for all used motors and the out of these values with linear regression determined no-load speed.

	experiment		data-sheet	
	stall torque [Nm]	no-load speed [rad/s]	stall torque [Nm]	no-load speed [rad/s]
MX 64	4.00	8.49	7.3	8.17
MX 106	6.80	6.82	10.0	5.76

Table 4.6.: The table shows a comparison between the determined stall torque and no-load speed and the values stated in the data sheets [O17, O16].

	mean static friction [Nm]
motor 1	0.14
motor 2	0.09
motor 3	0.04
all	0.09

Table 4.7.: The mean static friction values measured for the MX 64 motor are shown. For three motors, the static friction was measured at 12.8V, 14.8V, and 16.0V. The last column shows the mean value for all three motors together.

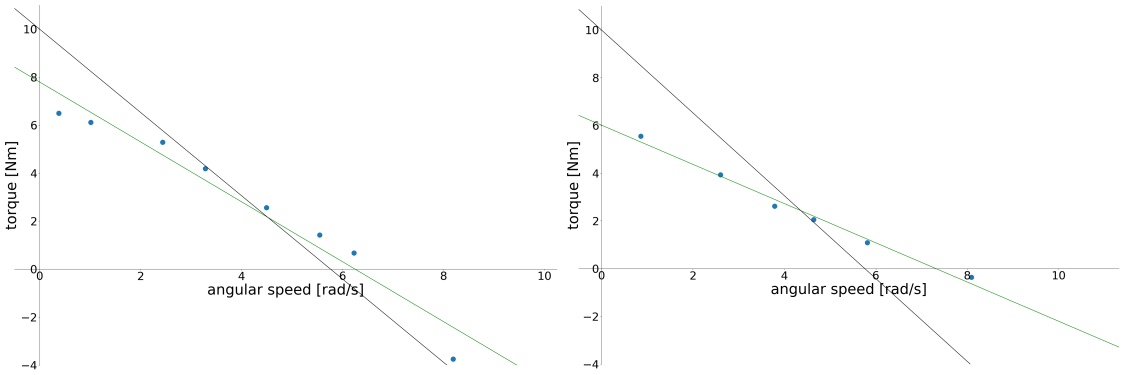


Figure 4.7.: The graph on the left shows the measurements of an old MX 106 motor with the blue dots representing the velocity values and their corresponding torque values per attached load. The green line is the regression line out of these dots, and the black line shows what is stated in the data-sheet. On the right the results from the experiments with a new MX 106 motor are shown the same way. Both measurements were done with 14.8 V as the input voltage.

	mean static friction [Nm]
motor 1	0.06
motor 2	0.06
motor 3	0.17
motor 4	0.01
all	0.10

Table 4.8.: The mean static friction values measured for the MX 106 motor are shown. For three motors, the static friction was measured at 12.8V, 14.8V, and 16.0V. The last column shows the mean value for all four motors together.

the MX 64 motors behaved quite similar independent of their age, there was a difference between the used and new MX 106 motors observable, which is illustrated in Figure 4.7. The new motor produced almost 2 Nm less torque. During the experiments this was also observable as the new motors could not lift a load of 2.5 kg anymore with all regarded voltage values, while the old motors stopped working at 3 kg with 14.8 V and 16V input and at 2.75 kg with 12.8 V.

The results of the static friction measurements are shown in Table 4.7 and 4.8. Overall the static friction is similar with 0.09 Nm for the MX 64 and 0.10 Nm for the MX 106. In comparison to the occurring torques during a motor's usage, it is small. With three or four motors, it is not possible to state the exact static friction coefficient. In Table 4.8, this is, for example, observable as the static friction values for motor three and four differ profoundly. Both motors have been used for the first time. Still, the results were different. This difference is explainable due to a different tightening of the motor horns, different gear mobility, and because of small offsets in the measurement, which occur despite biasing being used, which affect value in this small regime. However, overall, the mean values taken about all measurements of a motor type should be correct for an average specification. Therefore these values taken have been added to the URDF.

4.4. Summary

The URDF has been updated regarding its links and joints. For the links, the correct center of mass position and inertia tensor was determined. Furthermore, the correct mass value was stated. The joints description was improved regarding its physical parameters, too. The maximal speed and torque, as well as the static friction, were corrected. Moreover, the input voltage of the motors was studied, which showed that this is quite complex to model. It is known now that the input voltage value outputted from the motors is most accurate under medium load condition. However, the different motors vary quite a lot in their accurateness. Due to the torque and velocity experiments, an inadequacy in the stall torque and no-load speed values from the data-sheet was determined. Besides a difference between the new MX 106 motors and the already used ones was noted.

5. Evaluation

This chapter presents the evaluation of the improvements described in Chapter 4. It was investigated if the changes to the URDF helped to close the Reality Gap. For evaluation, different motions executed by the robot in simulation and reality were compared in terms of the position, velocity, and torque curves of all joints. In general, the evaluation is conducted with two animations the robot had to execute. The first is the cheering animation, shown in Figure 5.1. It is used to compare the movements of the arm joints and the behavior of the leg joints while they are staying at the same position. The second animation is the kicking animation, presented in Figure 5.2. This animation is useful to compare the behavior between reality and simulation under a high load condition in the right leg, as the robot is standing on this leg during the kick and also under a low load condition represented in the kicking left leg. The comparison between the simulation and reality is conducted with three recorded rosbags each. The three rosbags from reality are averaged, and the same is done with the three rosbags from the simulation. From the two resulting curves, the difference per time-step is taken for analysis. At first, just the improvements for the links were evaluated, which is shown in Section 5.1. After that, in Section 5.2, the new joint parameters were added to the URDF, and evaluation involving all improvements is performed. Finally, in Section 5.3, the results regarding the minimization of the Reality Gap are summarized.

5.1. Link Parameter Evaluation

The evaluation of the improvements of the links in the URDF is performed as described above. Furthermore, a quick head movement animation, as shown in Figure 5.3, is regarded to evaluate the inertia values from the URDF. This animation is used only for the link evaluation because inertia is only interesting for the link update in the URDF. The URDF contains for this evaluation the new parameters of the links. The joint values in the URDF are taken out of the data-sheet from the motors [O17, O16]. Hence, evaluating the link improvements is possible without influence by the new joint parameters.

At first, the cheering animation is evaluated in the context of the link improvements. In Figure 5.4b, the differences in the joints position curves, between the reality and the simulation using the link improved URDF is shown for all joints in the robot. Overall it stands out that the position curves of the head and arm joints have smaller differences than most of the leg joints. However, the differences in the arm joints are more distributed regarding the mean 50% values. Interestingly, the HipYaw joints are behav-

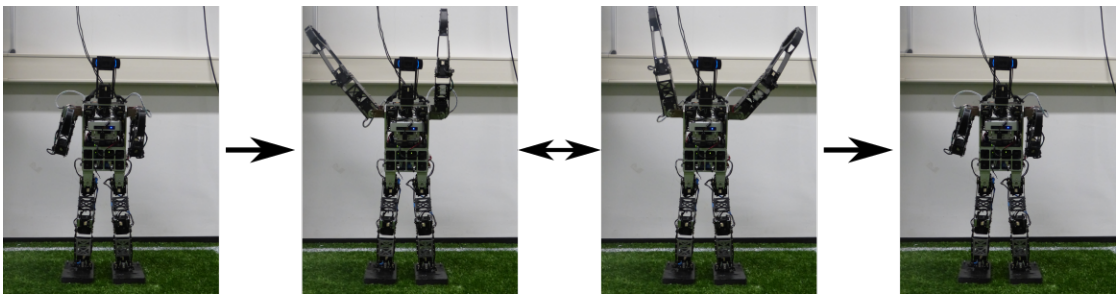


Figure 5.1.: The movement sequence of the cheering animation performed by a Wolfgang robot is shown. It is used to evaluate the Reality Gap regarding the motion behavior of the robot in simulation and reality.

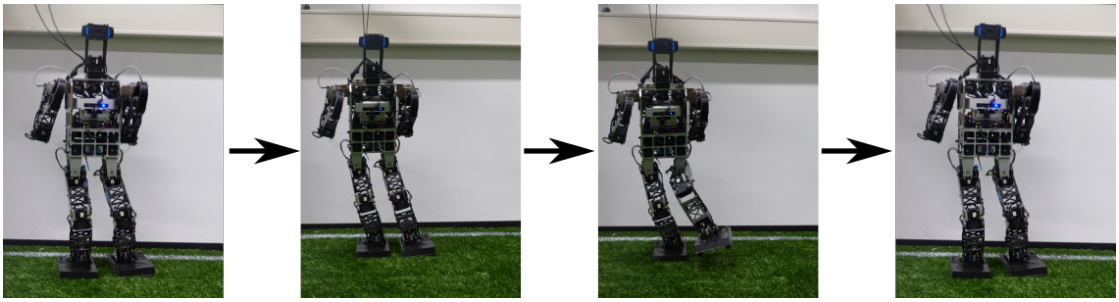


Figure 5.2.: The movement sequence of the kicking animation with the left foot performed by a Wolfgang robot is shown. It is used to evaluate the Reality Gap regarding the motion behavior of the robot in simulation and reality.

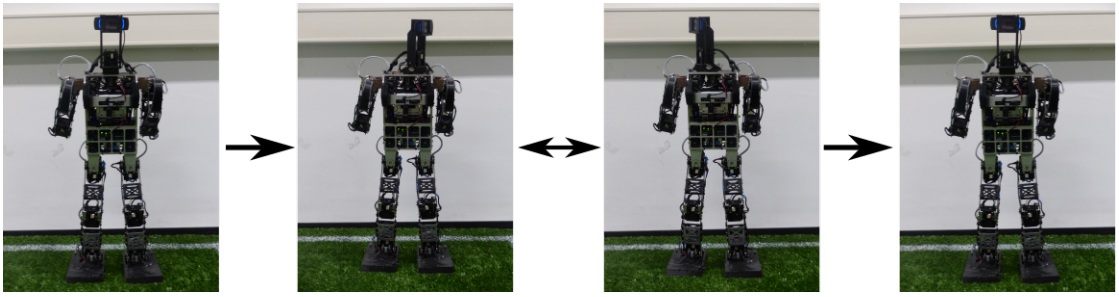


Figure 5.3.: The movement sequence of the head animation performed by a Wolfgang robot is shown. The head link moved several times from the left to the right side and back again. It is used to evaluate the Reality Gap regarding the inertia properties of the simulation.

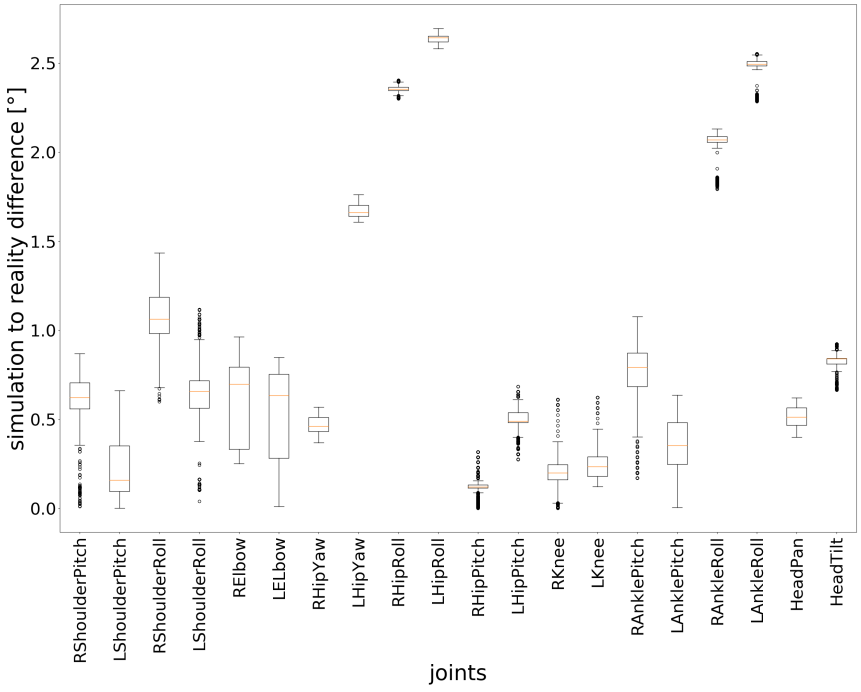
ing differently. The difference can be explained by taking a look at the position curves of these joints, as shown in Figure 5.7. The LHipYaw joints had an offset during the animation execution because of the standing position of the robot. Therefore a higher difference occurred compared to the simulation.

Looking at the related velocity curve differences, shown in Figure 5.5b, the not moving leg and head joints are almost accurate simulated. Some outliers appear because of the vibrations of the robot. The arm joints are slightly differing. Still, some outliers have a significant difference. The differences in the torque curves are shown in Figure 5.6b, which is more significant for not moving leg joints.

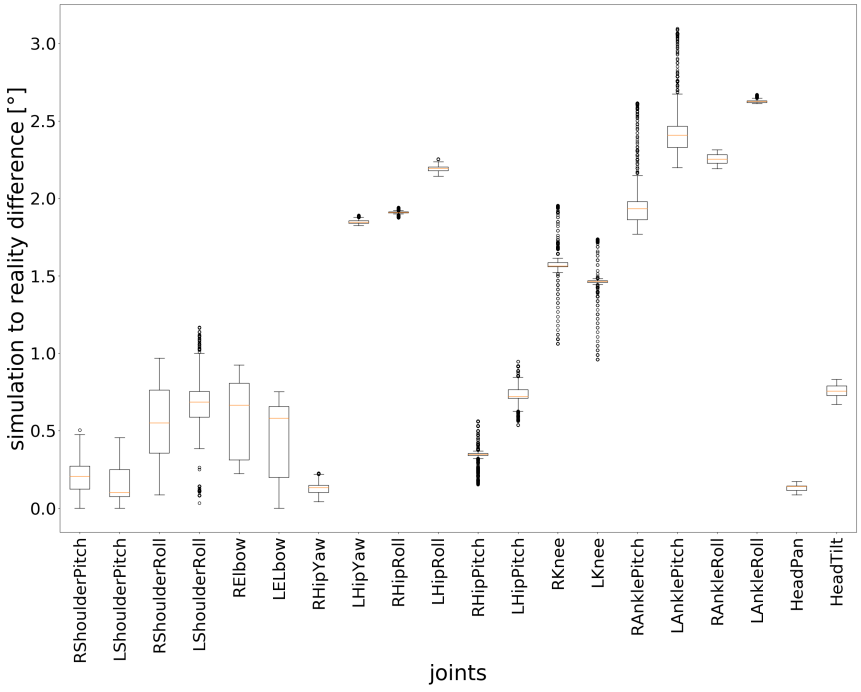
For evaluating if the link revision of the URDF improved the simulation of the robot, the above-described results are compared to the differences in the joint's movement curves between reality and simulation using the old URDF. The position differences analysis for both URDFs is visualized in Figure 5.4, the velocity differences analysis in Figure 5.5, and the analysis of the torque differences in Figure 5.6. These plots show that the simulation of the moving arm joints was improved because, e.g., the medians improved or kept a similar value. The position simulation of the not moving joints worsened a little, which is, e.g., represented in the box-plots of the position differences through higher median values and higher outlier ranges. The not moving joints simulation of the velocity achieves similar results using the link revised URDF compared to the old one. Just a few outliers range further. In the box-plots of the torque differences, it is shown for the not moving joints that the simulation of a few joints, e.g., RHipRoll and LHipRoll, got more accurate, while, for example, the joints RAnklePitch and LAnklePitch are simulated less precise. Overall this leads to a similar accuracy in simulation between the two URDFs regarding the torque simulation of the not moving joints. Table 5.1 confirms these results by showing the mean differences between simulation and reality for all moving and not moving joints. The mean position value of the moving arm joints improved about 0.17°, and the mean velocity improved slightly about $0.02^\circ/s$. The torque difference improved significantly about 1.5 Nm. The not moving leg joints got worse about 0.4° in the position difference, while the velocity and the torque difference stayed similar. The less accurate position simulation can be explained by stronger vibrations acting on the in simulation more massive torso after the link revision. So the PID controller has to counteract more to keep its position, which leads to a more imprecise simulation due to the not yet updated joint parameters.

The next animation regarded for evaluation is the above presented kicking animation. The position differences between reality and simulation using the link revised URDF are shown in Figure 5.8b. The simulation of the arms and the head is almost accurate. The leg joints position differ a little more. Interestingly, the distance of the quartiles of the right leg joints is bigger than the one in the left leg. Overall the medians for the right leg are also higher than the ones for the left leg. This is explainable due to the higher torques acting on the right leg, which is the one the robot is standing on.

Figure 5.9b shows the differences between the velocities. The velocity in the simulation is accurate except for lots of outliers in the curves from the leg joints. Because the

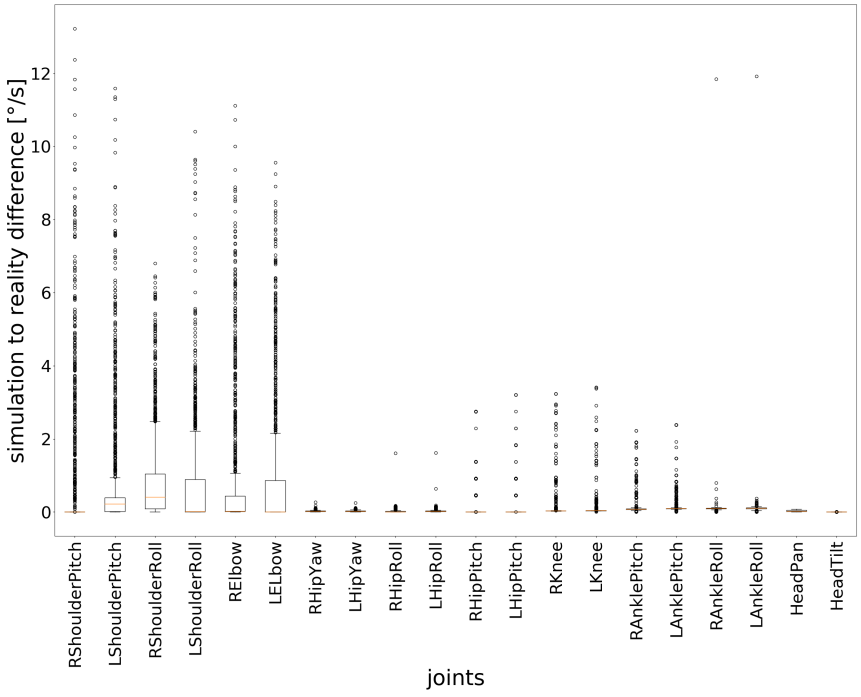


(a) old URDF

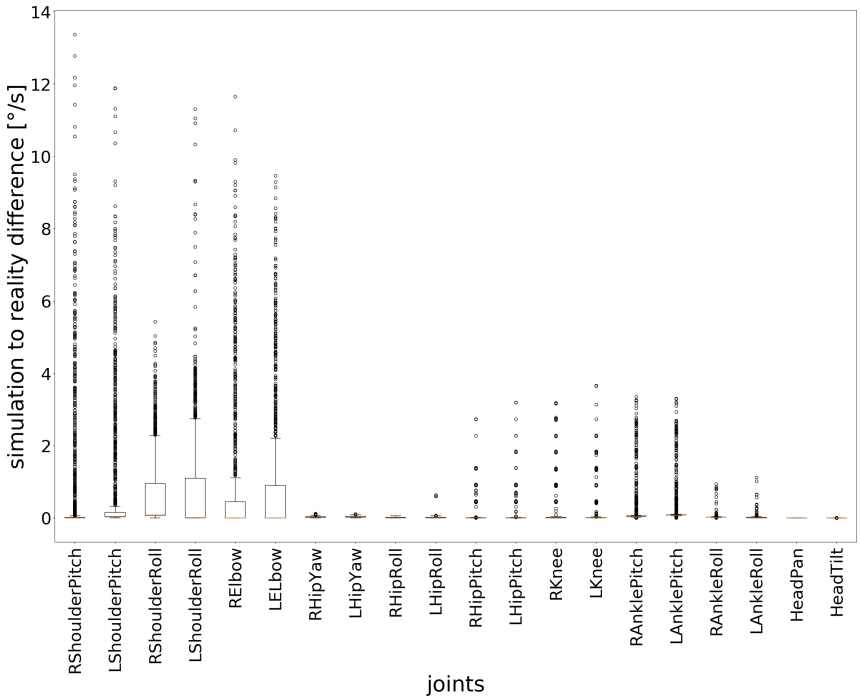


(b) link revised URDF

Figure 5.4.: These box-plots show the position differences of the cheering animation from the real robot and the simulated one using the old URDF on the top and the link revised URDF on the bottom. On the x-axis are the joints listed and on the y-axis is the difference in degrees plotted.

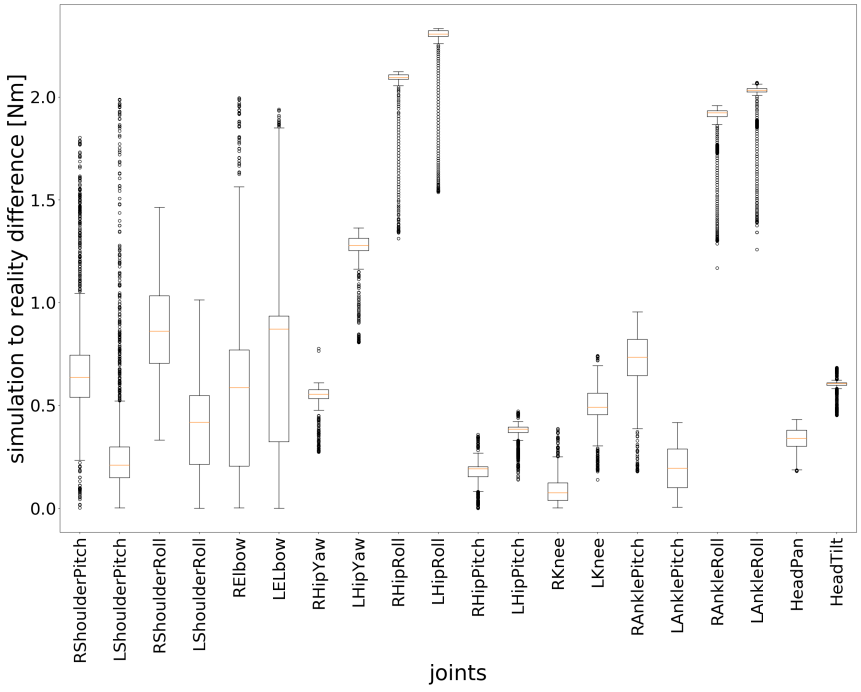


(a) old URDF

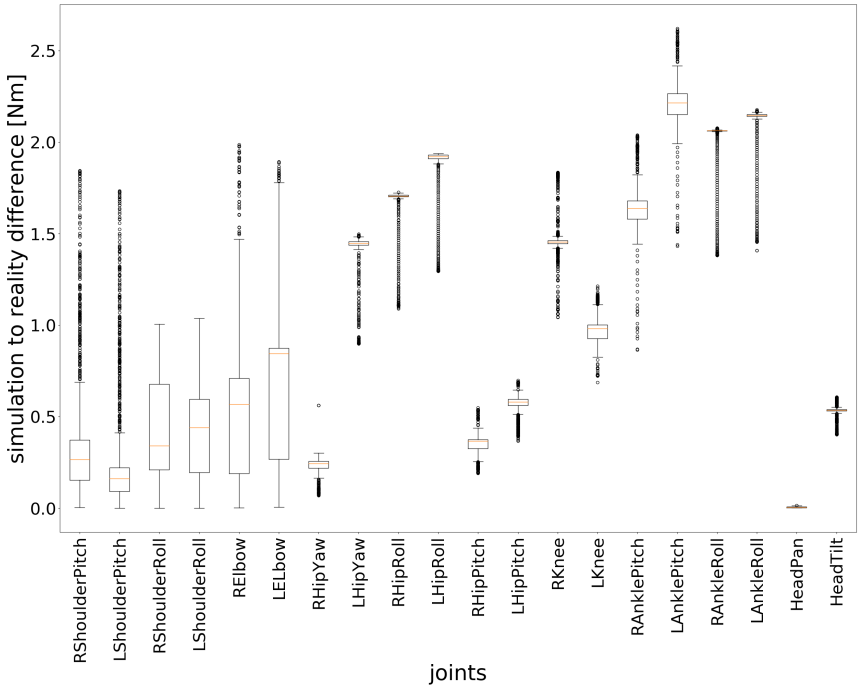


(b) link revised URDF

Figure 5.5.: These box-plots show the velocity differences of the cheering animation from the real robot and the simulated one using the old URDF on the top and the link revised URDF on the bottom. On the x-axis are the joints listed and on the y-axis is the difference in degrees per second plotted.



(a) old URDF



(b) link revised URDF

Figure 5.6.: These box-plots show the torque differences of the cheering animation from the real robot and the simulated one using the old URDF on the top and the link revised URDF on the bottom. On the x-axis are the joints listed and on the y-axis is the difference in Newton-meter plotted.

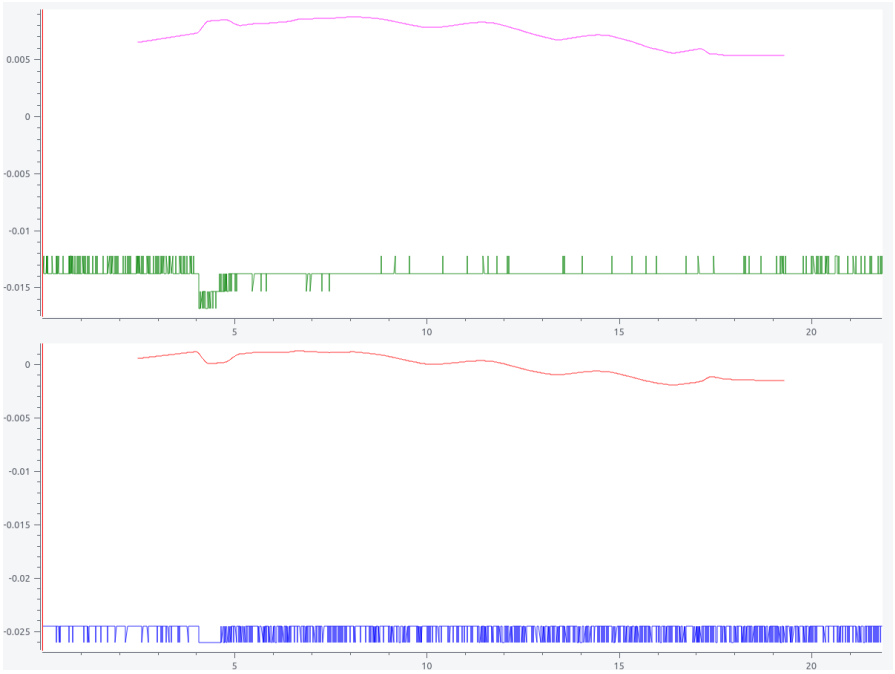
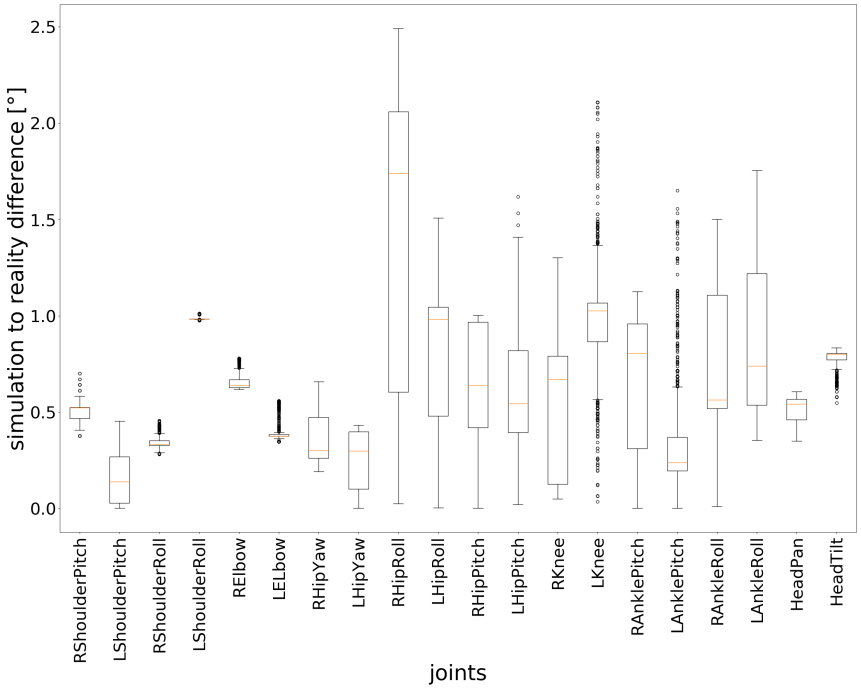


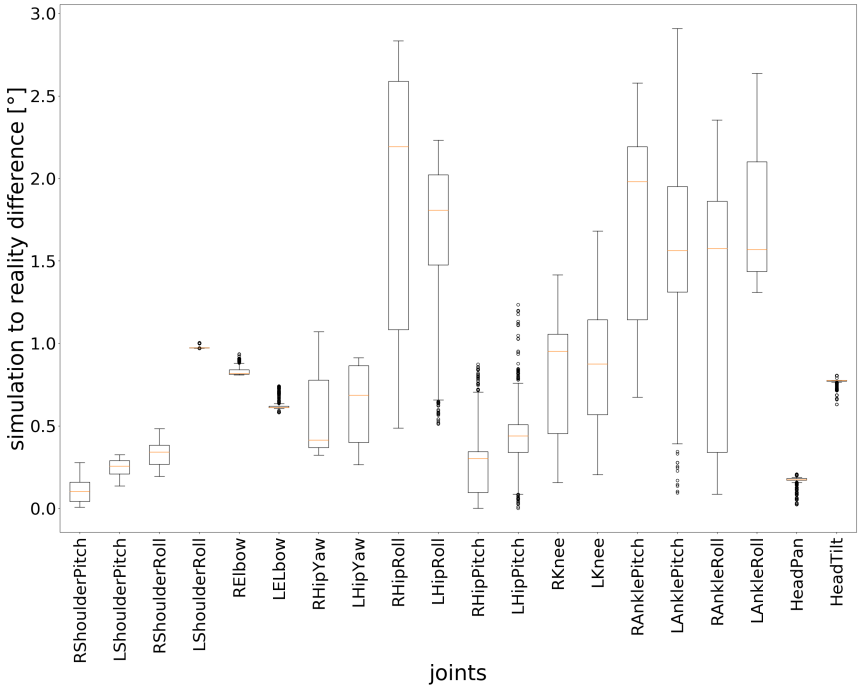
Figure 5.7.: This figure shows the position curves of the Hip Yaw joints during the cheering animation. On the top is the RHipYaw joint shown and on the bottom the LHipYaw joint. The green and blue graph display the joint position curves from the robot in reality. The two others show the curves recorded from simulation executed using the URDF after the link revision. On the x-axis the time is displayed in seconds. The y-axis shows the joint position in radians.

	position [°]	velocity [$°/s$]	torque [Nm]
moving, old URDF	0.6062	0.7271	1.9947
moving, link revised URDF	0.4305	0.7071	0.4445
unmoving, old URDF	1.0869	0.0643	0.9330
unmoving, link revised URDF	1.4582	0.0643	0.9331

Table 5.1.: The table shows the mean differences in position, velocity, and torque between the reality and the simulation using the old resp. link revised URDF during the cheering animation. The mean value is taken out of the mean differences from the single joints.

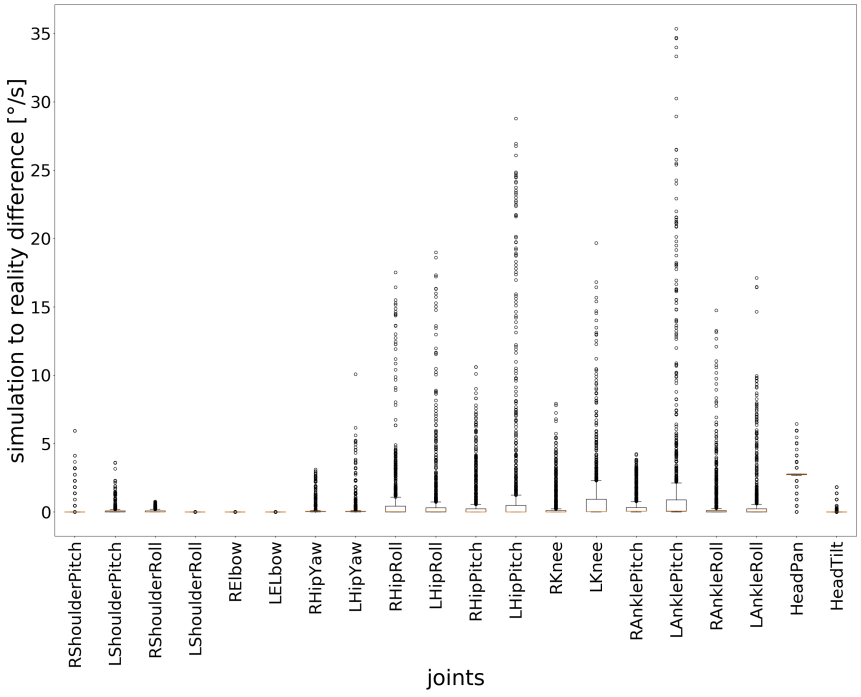


(a) old URDF

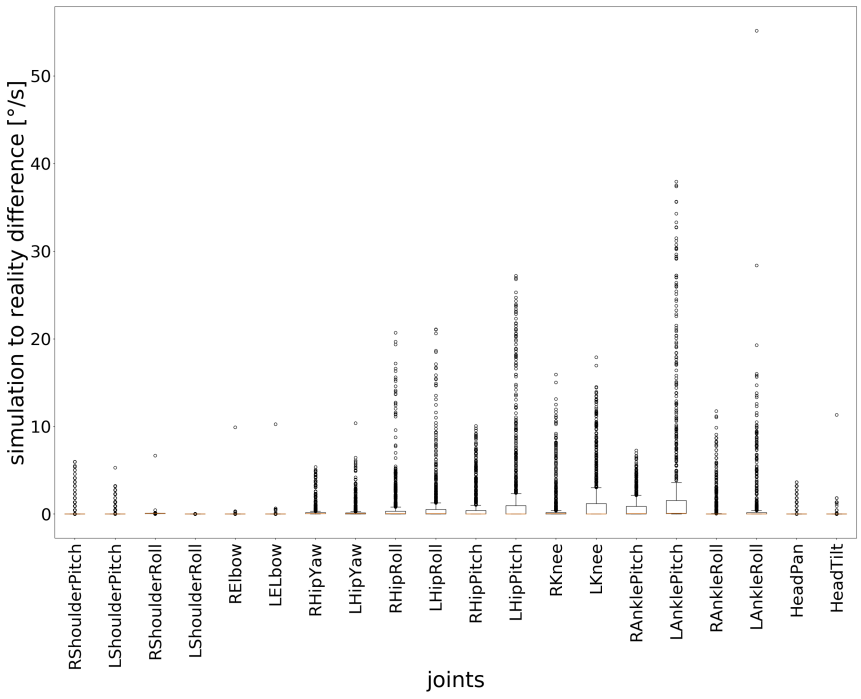


(b) link revised URDF

Figure 5.8.: These box-plots show the position differences of the kicking animation from the real robot and the simulated one using the old URDF on the top and the link revised URDF on the bottom. On the x-axis are the joints listed and on the y-axis is the difference in degrees plotted.

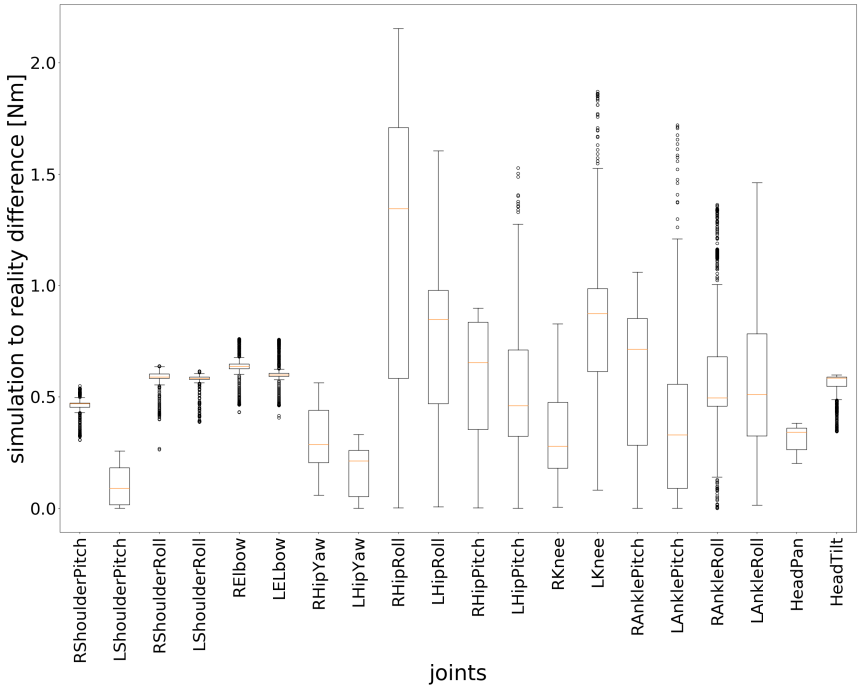


(a) old URDF

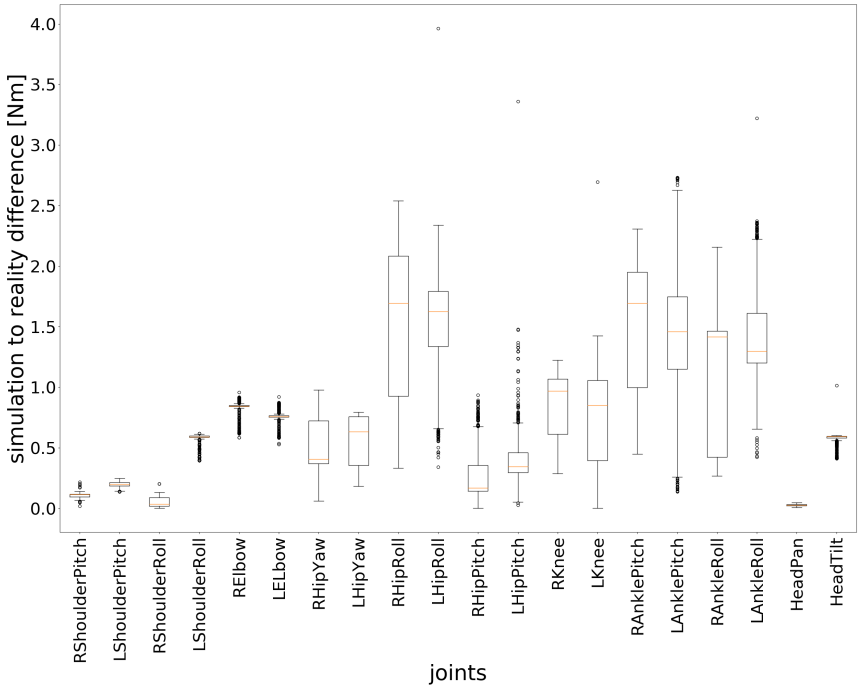


(b) link revised URDF

Figure 5.9.: These box-plots show the velocity differences of the kicking animation from the real robot and the simulated one using the old URDF on the top and the link revised URDF on the bottom. On the x-axis are the joints listed and on the y-axis is the difference in degrees per second plotted.



(a) old URDF



(b) link revised URDF

Figure 5.10.: These box-plots show the torque differences of the kicking animation from the real robot and the simulated one using the old URDF on the top and the link revised URDF on the bottom. On the x-axis are the joints listed and on the y-axis is the difference in Newton-meter plotted.

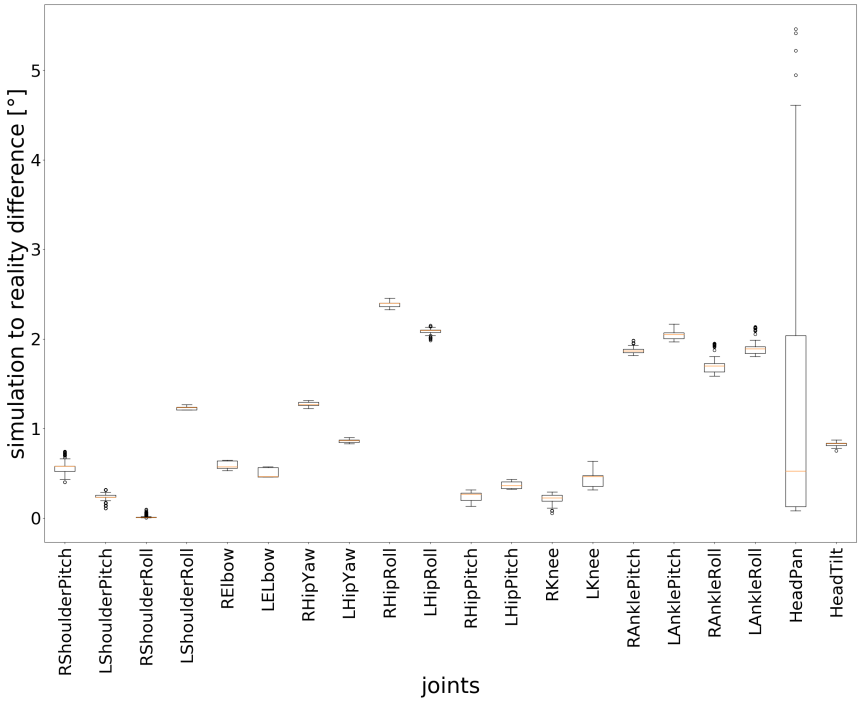
	position [°]	velocity [$^\circ/s$]	torque [Nm]
moving, old URDF	0.6917	0.7325	0.5710
moving, link revised URDF	1.1325	0.9631	0.9993
unmoving, old URDF	0.5446	0.3859	0.4841
unmoving, link revised URDF	0.5064	0.0665	0.3949

Table 5.2.: The table shows the mean differences in position, velocity, and torque between the reality and the simulation using the old resp. link revised URDF during the kick animation. The mean value is taken out of the mean differences from the single joints.

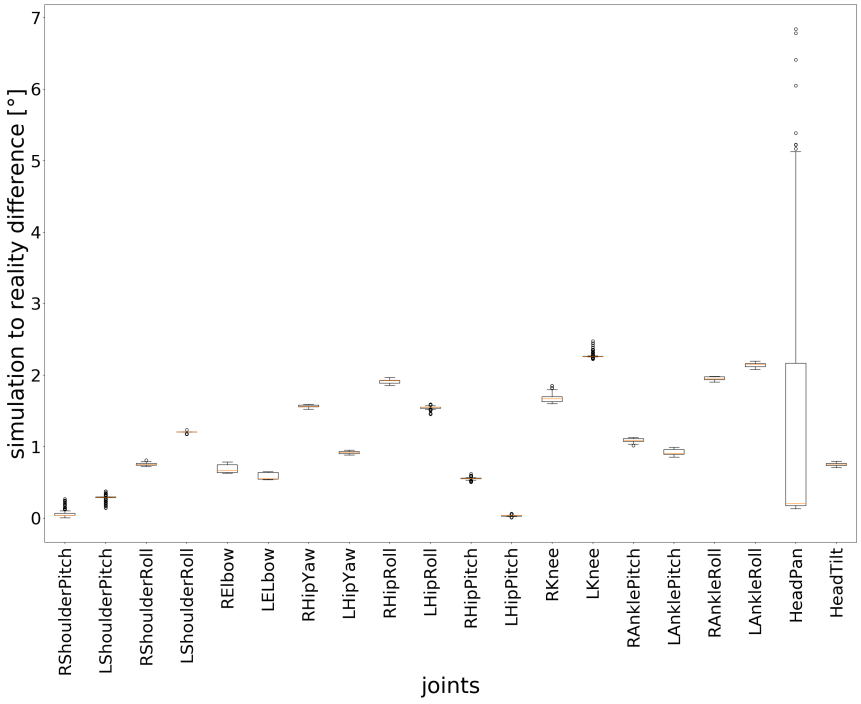
left leg is the moving one in the regarded animation, the outlier range is higher in that leg. The difference in the torque values is shown in Figure 5.10b. The simulation of the joints torque is imprecise for most of the leg joints, while again the right leg is less accurate.

Comparing the above-shown results from the kicking animation with the differences between reality and the old URDF a similar result as with the cheering comparison is observable. Figure 5.8 shows that the differences in position in the arm and head joints again improved, which means that the simulation got closer to reality. However, the leg joints differences again mostly decreased a little. The difference in the velocity curves improved for the arm and head joints. This improvement is shown in Figure 5.9, where, e.g., the HeadPan joint has a smaller median. For the leg joints, the box-plots show significantly higher outliers in the plot with the link revised URDF. The torque differences are compared in Figure 5.10. Again an improvement is represented for the unmoving arm and head joints. For example, the median torque difference of the RShoulderPitch and HeadPan joints decreased. The leg joints difference worsened, which is, e.g., presented in the higher median of the LAnklePitch joint. In Table 5.2 are the mean difference values for position, velocity, and torque stated from the comparison from reality to the old URDF, as well as reality to the link revised URDF. The arm joints simulation of position improved in average about 0.04°, which is a smaller improvement than for the cheering animation. The mean velocity difference improved for the arm joints again about 0.3$^\circ/s$. The torque of the arm and head joints also improved. The worsening of the leg joints values is also shown in the table. A possible reason for that is the influence of incorrect joint values on the corrected link parameters. This influence consists of the PID controller and outside forces affecting the more massive robot stronger because the simulation of the leg joints which are moving or under high load is more imprecise while the simulation of the arm joints is not.

As inertia has a higher effect on fast motions, a fast head movement is evaluated too. The animation is executed so fast that the robot could not reach the goal position in the limited time per keyframe. The position accuracy, shown for the old and link revised URDF in Figure 5.11, improved for the moving HeadPan joint as the median in the difference decreased. Furthermore, some not moving joints were more accurate after

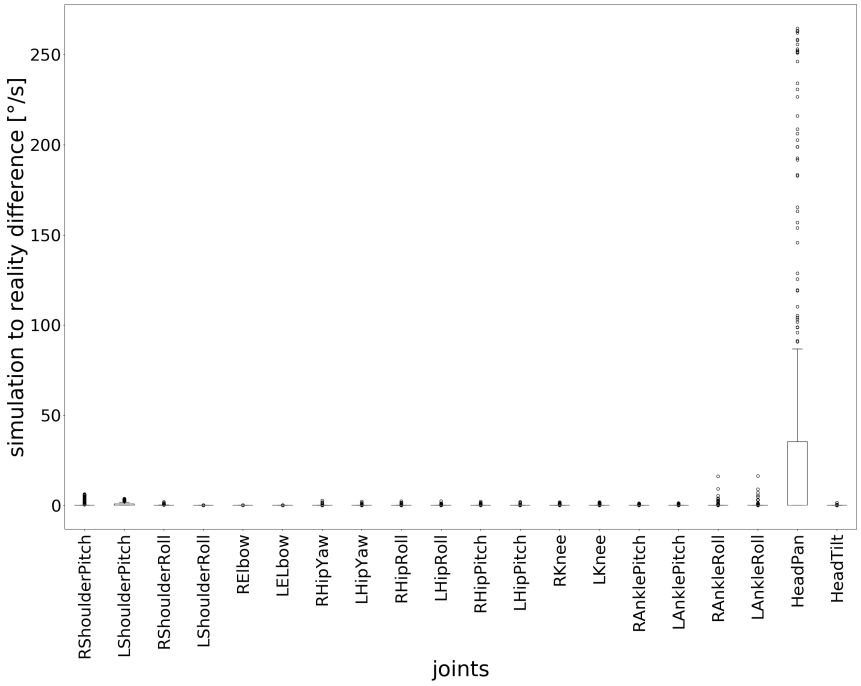


(a) old URDF

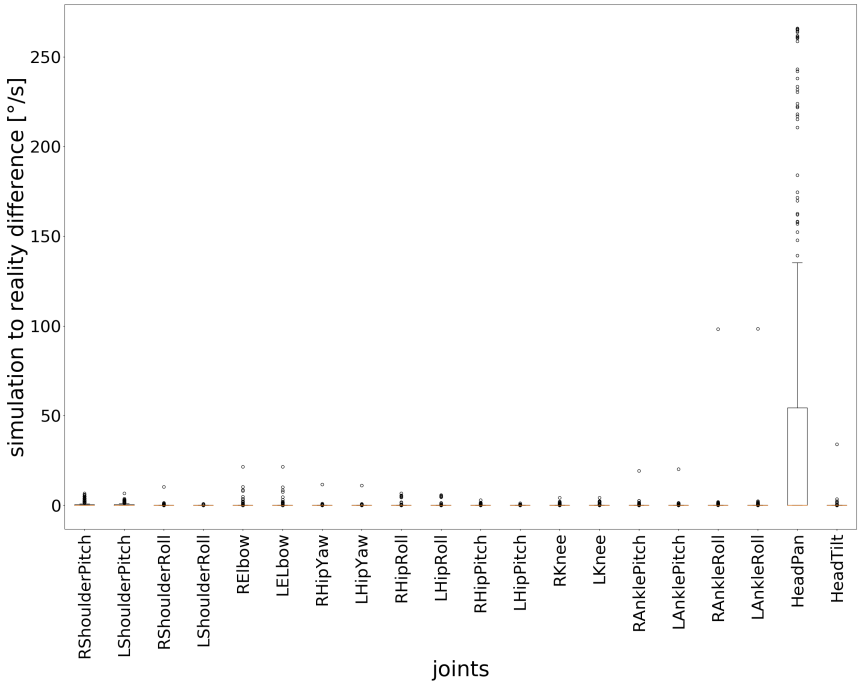


(b) link revised URDF

Figure 5.11.: These box-plots show the position differences of the head movement animation from the real robot and the simulated one using the old URDF on the top and the link revised URDF on the bottom. On the x-axis are the joints listed and on the y-axis is the difference in degrees plotted.

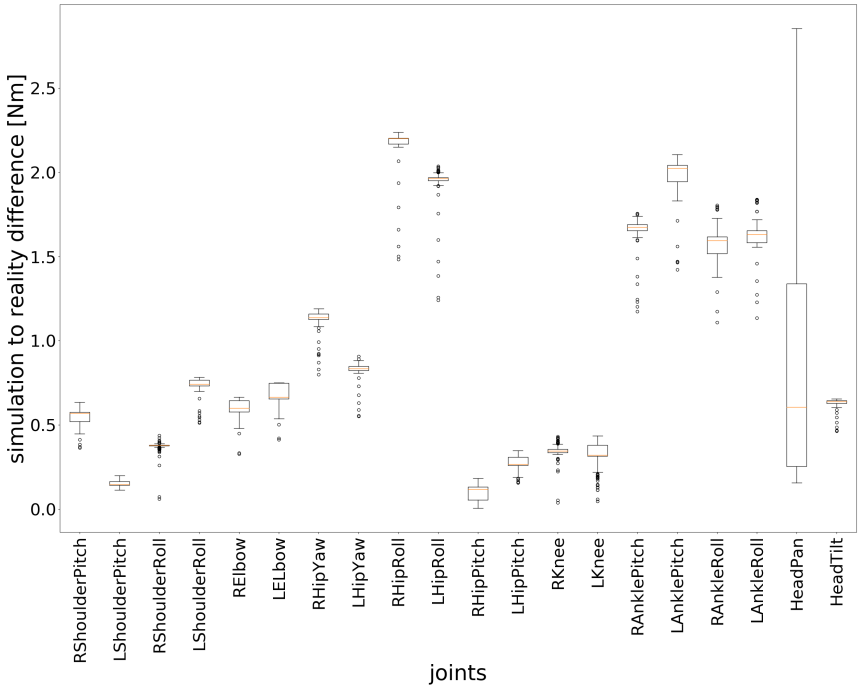


(a) old URDF

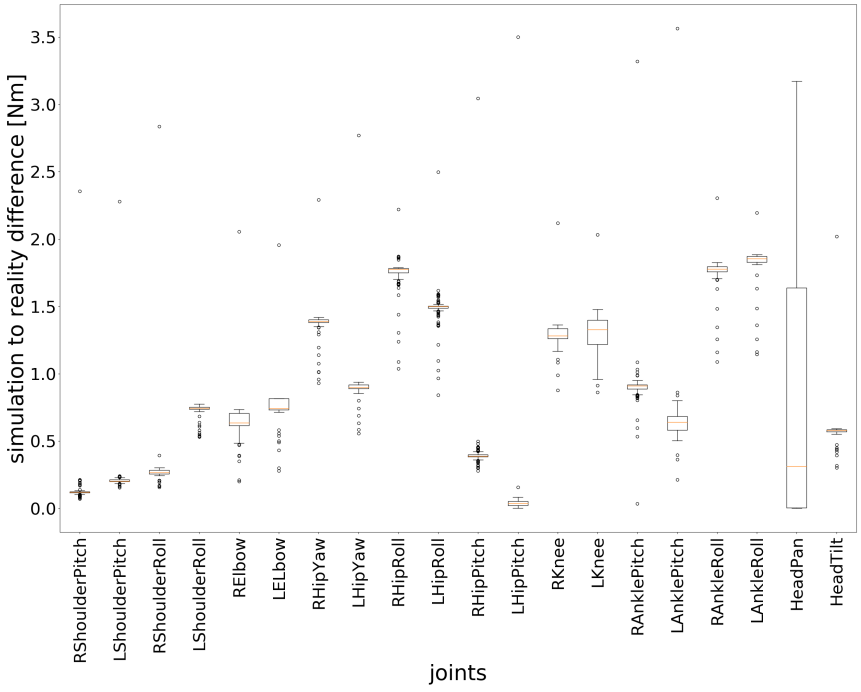


(b) link revised URDF

Figure 5.12.: These box-plots show the velocity differences of the head movement animation from the real robot and the simulated one using the old URDF on the top and the link revised URDF on the bottom. On the x-axis are the joints listed and on the y-axis is the difference in degrees per second plotted.



(a) old URDF



(b) link revised URDF

Figure 5.13.: These box-plots show the torque differences of the head movement animation from the real robot and the simulated one using the old URDF on the top and the link revised URDF on the bottom. On the x-axis are the joints listed and on the y-axis is the difference in Newton-meter plotted.

	position [°]	velocity [$^\circ/s$]	torque [Nm]
moving, old URDF	1.1258	39.5268	0.8825
moving, link revised URDF	1.2376	44.4872	0.7342

Table 5.3.: The table shows the mean differences in position, velocity, and torque between the reality and the simulation using the old resp. link revised URDF during the head movement animation. Just the moving HeadPan joint is shown as this is the interesting one when analyzing inertia. The mean value is taken out of the mean differences from the single joints.

the link revision. An example of this are the HipRoll joints. The plots for velocity (Figure 5.12) show an improvement in the moving joint as the upper quartile improve about 50%. However, overall, there are more outliers in the velocity difference analysis box-plot visualizing the link revised data. The torque difference (Figure 5.13) improved, which can be noticed at the moving joint whose median for example got smaller. While some not moving joints like the AnklePitch joints also improved, there exist more high outliers after the link revision in the torque difference box-plot.

The inertia update was helpful for a more accurate torque simulation, which can be seen in Table 5.3. The position and velocity accuracy decreased a little, possibly, because of wrong motor speeds, due to imprecise values for the PID controller, which lead to a wrong simulation of the inertia.

5.2. Complete Evaluation

In this section, it is evaluated how effective the joint parameter improvement was to minimize the Reality Gap of the Wolfgang robot. For evaluation, the, at the beginning of the chapter explained, cheering and kicking animations are used. The URDF contains now the new velocity, torque, and friction parameters for the joints, as well as the new inertial link parameters. The evaluation is conducted by a comparison of the results achieved with the new URDF to the results from the evaluation using only the new inertial link parameters in the URDF and the joint values from the data-sheet.

Again the cheering animation is analyzed at first. In Figure 5.14b, the position differences from the simulation using the new URDF and reality are presented. The position is almost accurate simulated as the worst difference is approximately 1.43°. Overall the difference between simulation and reality of the arm and head joints is smaller than the difference of the leg joints. The median velocity difference, displayed in Figure 5.15b, is also near zero, which symbolizes an accurate simulation. Just some outliers, especially in the arm joints, still differ a lot from the in reality executed motion curve. The difference in the torque curves, which is presented in Figure 5.16b, is still differing. Especially the leg joints have on average a difference of 1 Nm to the real curve.

	position [°]	velocity [$^\circ/s$]	torque [Nm]
moving, old URDF	0.6062	0.7271	1.9947
moving, link revised URDF	0.43048	0.7071	0.4445
moving, new URDF	0.3138	1.6838	0.2798
unmoving, old URDF	1.0869	0.0643	0.9330
unmoving, link revised URDF	1.4582	0.0643	0.9331
unmoving, new URDF	0.6918	0.2330	0.5792

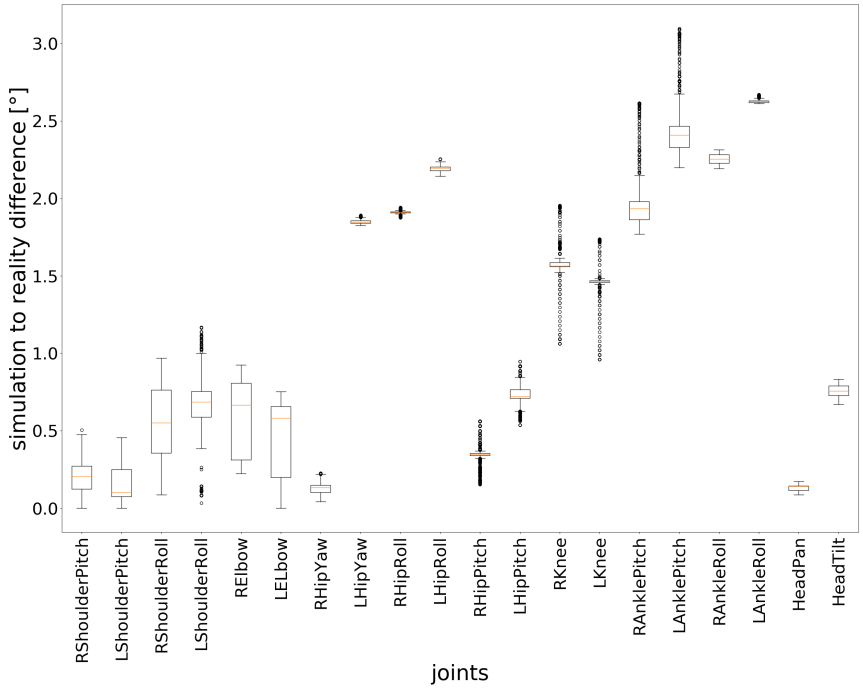
Table 5.4.: This table states the mean differences between the simulation using the named URDF and the measurements of the real cheering motion execution. Thereby the mean difference for position, velocity, and torque of the mean differences of each joint is shown.

	position [°]	velocity [$^\circ/s$]	torque [Nm]
moving, old URDF	0.6917	0.7325	0.5710
moving, link revised URDF	1.1325	0.9631	0.9993
moving, new URDF	0.5606	1.0626	0.4622
unmoving, old URDF	0.5446	0.3859	0.4841
unmoving, link revised URDF	0.5064	0.0665	0.3949
unmoving, new URDF	0.3625	0.0651	0.1702

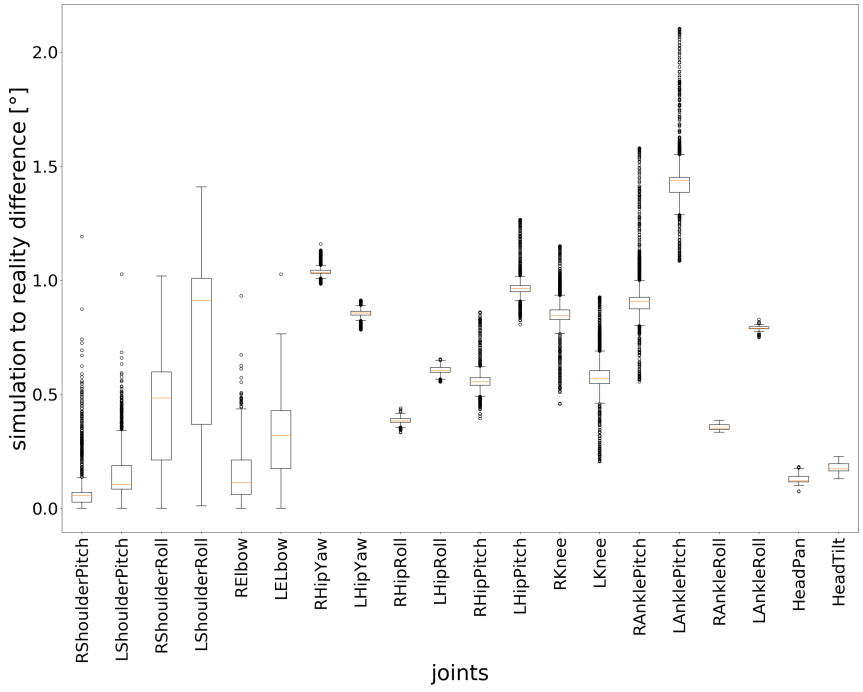
Table 5.5.: This table states the mean differences between the simulation using the named URDF and the reality of the kicking motion execution. Thereby the mean difference for position, velocity, and torque of the mean differences of each joint is shown.

In comparison to the differences between the link revised URDF and reality, shown in Figure 5.14, the simulation of the joint positions improved. For example, the LAnklePitch joint got approximately 8.6° closer to the real position curve. The comparison of the velocity differences (Figure 5.15) shows that while the median values are still equal, there are more values, using the new joint parameters, deviating from the real motion velocity. Therefore the velocity simulation accuracy decreased a little. After the joint parameters update, the torque values are more accurate for all motors now. Merely some arm joints have some higher outliers than it was the case before the joint update. The analysis is confirmed by the mean difference taken over all joints mean differences shown in Table 5.4. Position and torque are, using the new URDF, more accurately simulated, while the velocity simulation of the moving arm joints got significantly worse because of all the outliers. The decrease of the velocity simulation accuracy of the other joints is also represented.

Next, the kicking animation is evaluated with the new URDF. The position, shown in Figure 5.17b, is almost accurately simulated because the highest medians are slightly bigger than 1°. The medians in the velocity box-plot (Figure 5.18b) are near zero. However, there are lots of high outliers, especially for the leg joints. The torque medians

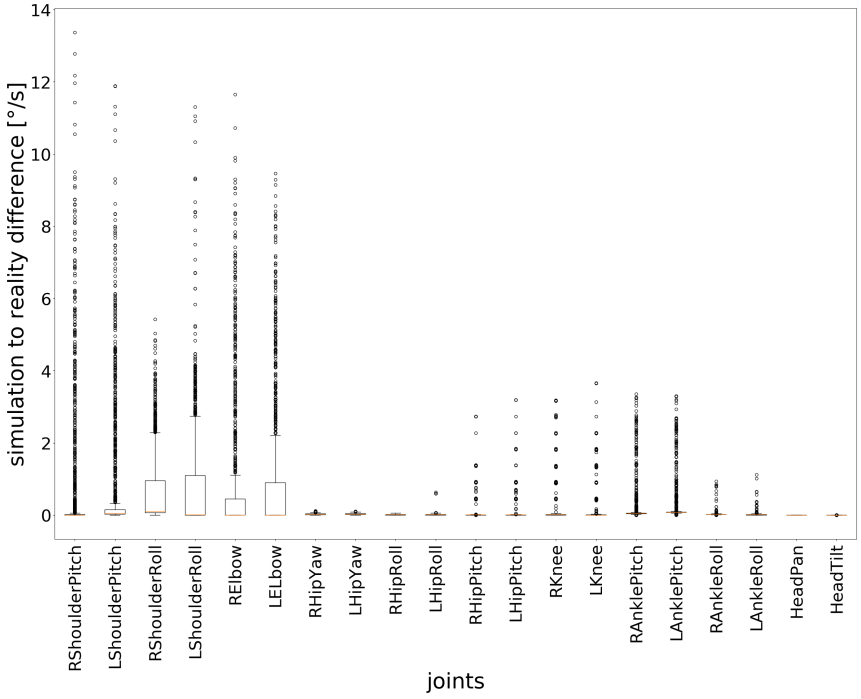


(a) link revised URDF

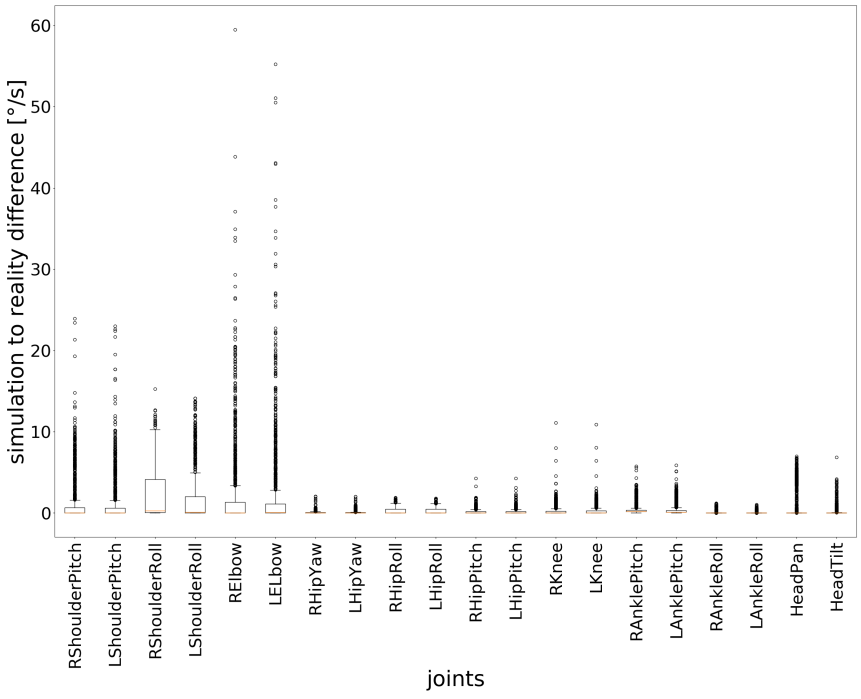


(b) new URDF

Figure 5.14.: These box-plots show the position differences of the cheering animation from the real robot and the simulated one using the link revised URDF on the top and the new URDF on the bottom. On the x-axis are the joints listed and on the y-axis is the difference in degrees plotted.

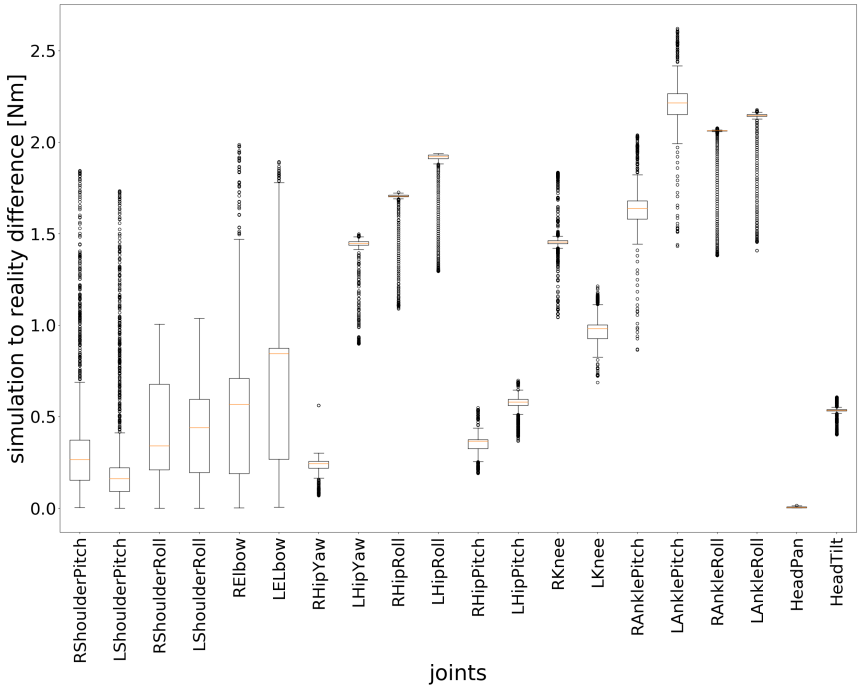


(a) link revised URDF

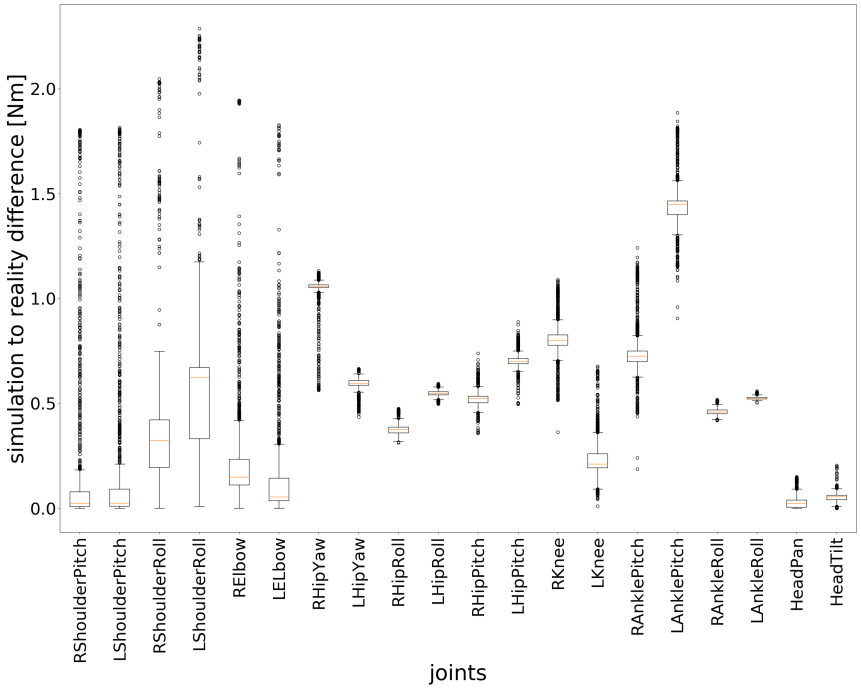


(b) new URDF

Figure 5.15.: These box-plots show the velocity differences of the cheering animation from the real robot and the simulated one using the link revised URDF on the top and the new URDF on the bottom. On the x-axis are the joints listed and on the y-axis is the difference in degrees per second plotted.

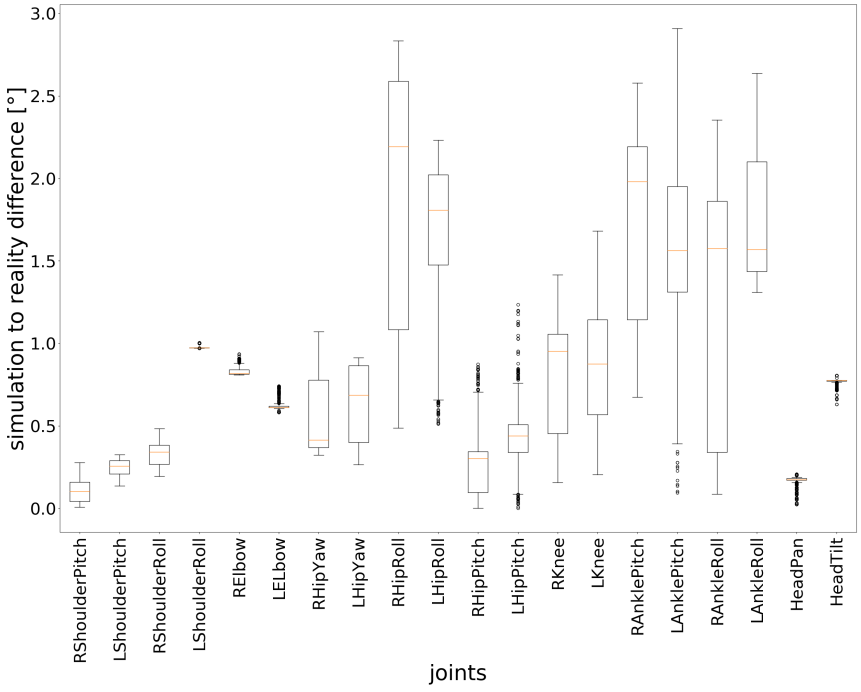


(a) link revised URDF

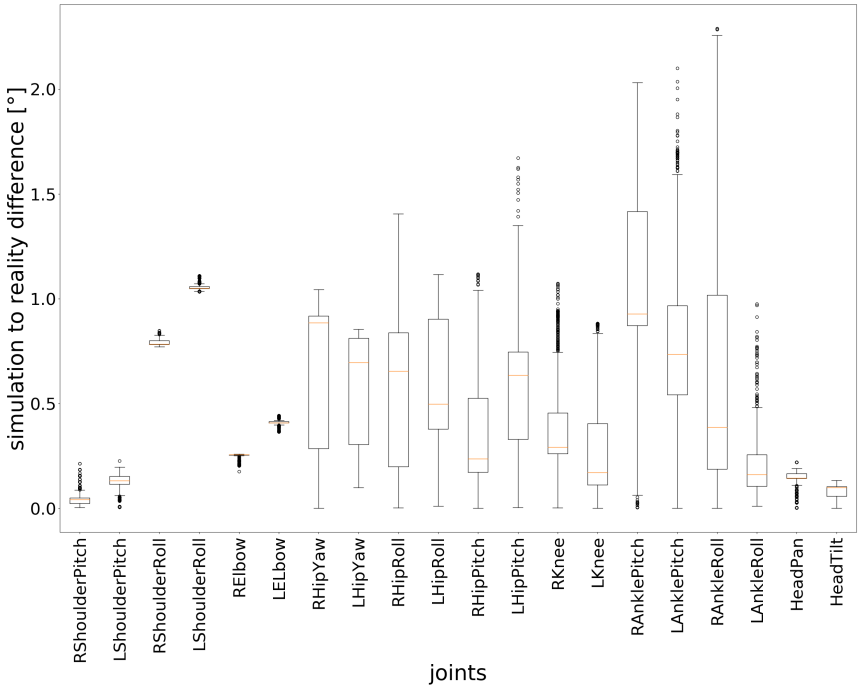


(b) new URDF

Figure 5.16.: These box-plots show the torque differences of the cheering animation from the real robot and the simulated one using the link revised URDF on the top and the new URDF on the bottom. On the x-axis are the joints listed and on the y-axis is the difference in Newton-meter plotted.

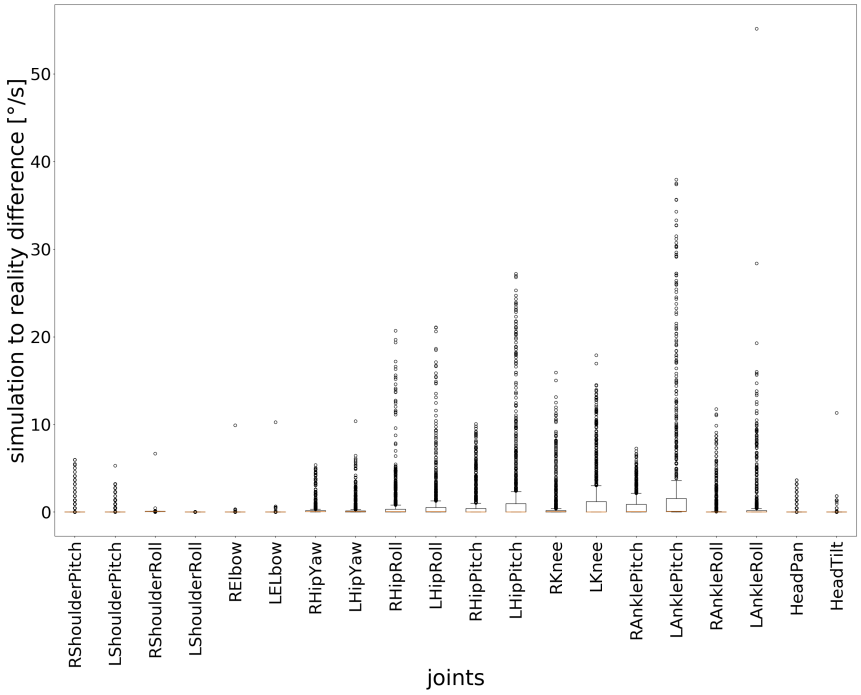


(a) link revised URDF

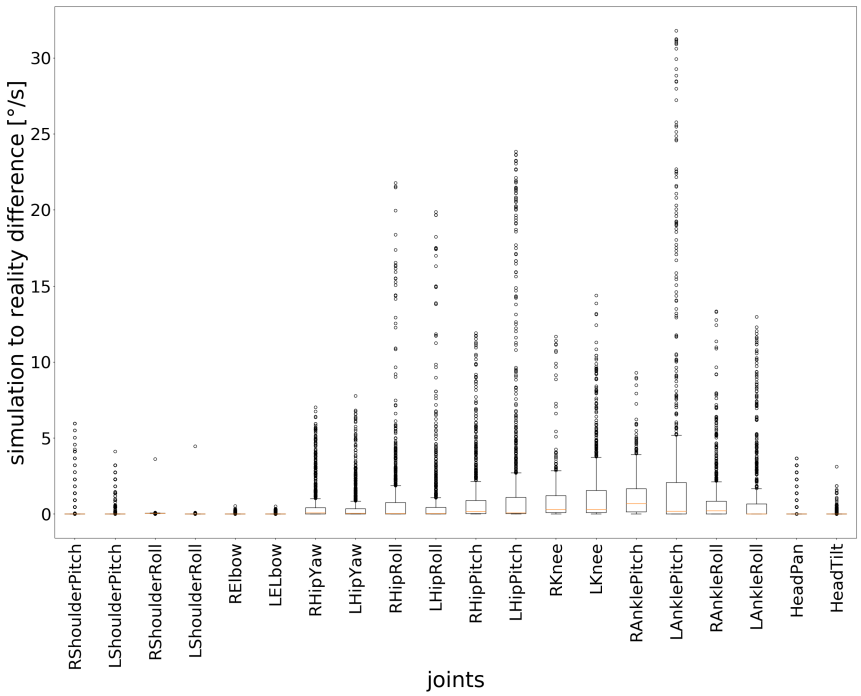


(b) new URDF

Figure 5.17.: These box-plots show the position differences of the kicking animation from the real robot and the simulated one using the link revised URDF on the top and the new URDF on the bottom. On the x-axis are the joints listed and on the y-axis is the difference in degrees plotted.

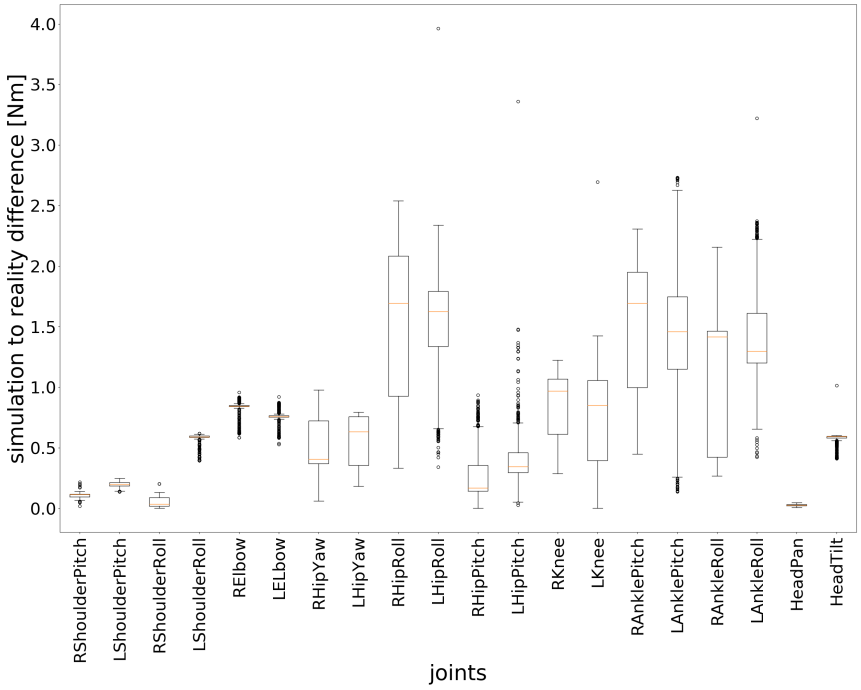


(a) link revised URDF

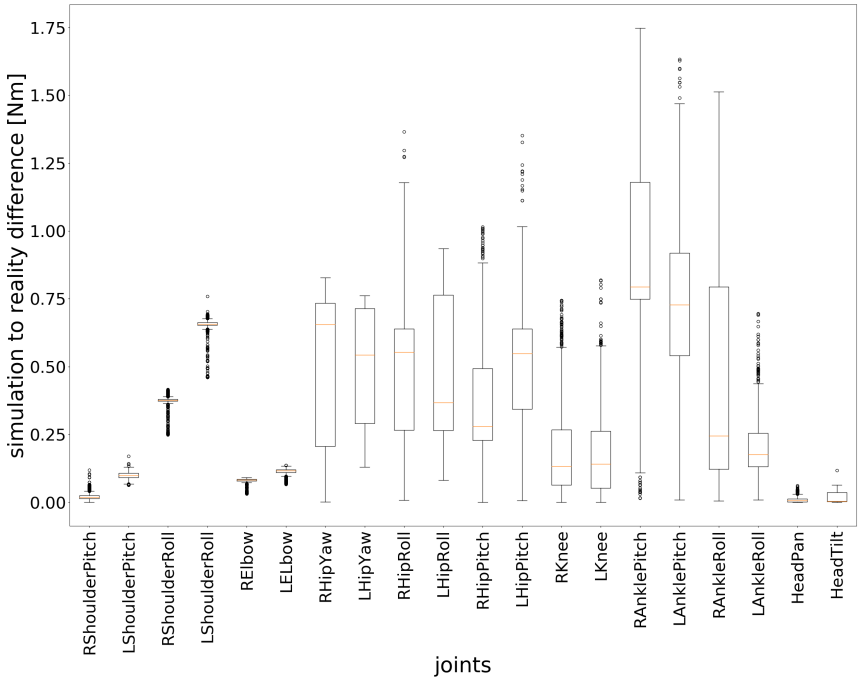


(b) new URDF

Figure 5.18.: These box-plots show the velocity differences of the kicking animation from the real robot and the simulated one using the link revised URDF on the top and the new URDF on the bottom. On the x-axis are the joints listed and on the y-axis is the difference in degrees per second plotted.



(a) link revised URDF



(b) new URDF

Figure 5.19.: These box-plots show the torque differences of the kicking animation from the real robot and the simulated one using the link revised URDF on the top and the new URDF on the bottom. On the x-axis are the joints listed and on the y-axis is the difference in Newton-meter plotted.

differ in maximum about 1 Nm, which is presented in Figure 5.19b. It symbolizes an almost accurate simulation for the torque.

Comparing these results to the results from using just the link revision, an improvement in the position simulation is visible, which is shown in Figure 5.17. For example, the highest median in the link revised box-plot is halved in the box-plot with the new URDF due to the new joint parameters. The velocity differences, presented in Figure 5.18, are similar. The differences in the torque values (Figure 5.19) overall got better because the highest median after the link revision was around 1.75 Nm, while the highest one after the joint revision is smaller than 1 Nm. The comparison of the mean differences of all joints mean differences, presented in Table 5.5 confirms the improvements for the position and torque too. While the velocity accuracy is similar or just a little worse in comparison to the one using the link revised URDF, a bigger difference is visible to the old URDF. An explanation could be the complexity of the simulation modeling as it is affected by outside forces and the PID controller.

5.3. Summary

The result of the evaluation is presented in Table 5.6. The changes in the URDF, lead to an improvement in the simulation of the joints position and torque. However, the simulation of the joints velocity got less accurate except for the unmoving joints during the kicking animation. It was figured out that it is not helpful to only adapt the link parameters of the robot as this leads to a worsening of the simulation using the wrong joint parameters. Together with the joint parameter update this then leads to the mentioned changes. The position and the torque simulation do not map the reality exactly yet, but significant improvements to the accuracy could be achieved. The velocity simulation still needs to be improved. Adapting the parameters in the simulation is not enough as this is, for example, profoundly affected by the PID control algorithm and outside forces acting on the robot. These two aspects do not influence the unmoving joints in the kicking animation a lot, which resulted in an improvement of their velocity accuracy, while the simulation of the other joints, and during movement of these joints, decreased.

	position [°]	velocity [$°/s$]	torque [Nm]
cheering, moving joints	0.2924	-0.9567	1.7149
cheering, unmoving joints	0.3951	-0.1687	0.3538
kicking, moving joints	0.1311	-0.3301	0.1088
kicking, unmoving joints	0.1821	0.3208	0.3139

Table 5.6.: This table shows the result of the evaluation. The mean accuracy of the old URDF is compared to the mean accuracy of the new URDF. Presented is the result through the difference between the position, velocity, and torque values. The difference illustrates the resulting changes due to the, in this thesis, made improvements to the URDF. The difference being a positive number symbolizes an increase in the simulation accuracy, while a negative number symbolizes a decrease.

6. Discussion

In Chapter 5, it was figured out that the simulation with the improved URDF led to a more accurate simulation of the position and torque of the robot's joints, while the velocity accuracy decreased. With this result, the simulation usage for checking a motion behavior of the robot is more efficient because the position curves are closer to reality. To mimic the robot's behavior, in reality, it is necessary to improve the simulation of the joint velocities, especially. It is not possible to realize this through only updating the parameters in the URDF, as the velocity depends highly on the PID control algorithm. A difference in the PID control is indicated in simulation because the motors need a long time to move the last degree before the goal position is reached, which does not happen in reality under normal load conditions. Ideally, the PID parameters of the robot should be used in the simulation. At the current state of the simulation, this is not possible as the robot is not able to stand in simulation with the PID parameters used on the real robot. Further investigation is needed to explain this. The method of operation of the PID controller in the real motor is shown in Figure 6.1, which visualizes the position control mode of a Dynamixel MX series motor [O39]. The P, I and D part are summed up, while being weighted with a coefficient and time. That represents the usual realization of a PID controller. The position control of the used motors can be adapted with a feed-forward controller, but this feature is not used in the motors, which were regarded in this thesis. In the simulation the PID controller is implemented in the used *ros_control* controller [O13]. The PID control is implemented in the usual way. Therefore this is similar to the PID controller used in the real motors. The simulated controller regards the time passed since the last update, so there cannot be a problem with too small update rates of the simulation. Therefore it is not known why the simulation is not working with the on the real robot used PID values. A further aspect influencing the simulation is the not yet modeled dynamic friction, as modeling requires complex setups to measure the different components of the dynamic friction. All the mentioned aspects have an impact on the Reality Gap.

The robot's behavior in simulation is influenced by lots of different factors which were not studied in this thesis. One factor is that the ground the robot is standing on is simulated flatly. In reality, the robot is standing on artificial grass. This effects, for example, how stable the robot is standing, or collision forces with the ground. Therefore an analysis, like in Chapter 5, was made between a cheering animation execution in simulation using the new URDF and an execution in reality on flat ground. The results are presented in Table 6.1. It is shown that the simulation is more accurately mimicking the robot's behavior on the grass and not on the flat ground. The position of the joints was 0.2° less

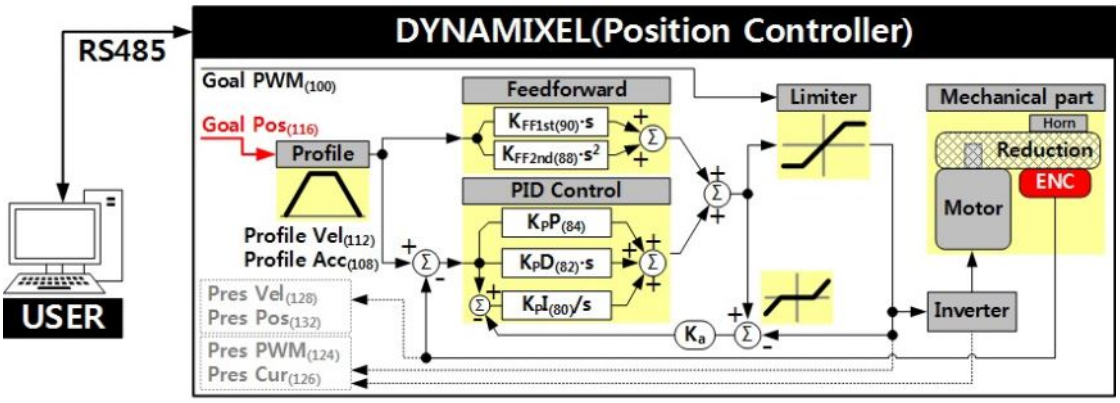


Figure 6.1.: The position controller of a Dynamixel MX series motors is visualized. It is shown how the motors are actuated. Furthermore, the for this thesis relevant calculation of the PID controller output is shown. This figure is taken from the Robotis e-Manual for this motor type [O39].

	position [°]	velocity [$^\circ/s$]	torque [Nm]
moving, flat ground	0.3345	0.6742	1.6723
moving, grass	0.5606	1.0626	0.4622
unmoving, flat ground	0.5512	0.2366	0.4414
unmoving, grass	0.3625	0.0651	0.1702

Table 6.1.: The table shows the mean of the mean differences of the single motors in position, velocity, and torque between the comparison of cheering animation simulated using the new URDF and executed in reality on grass respectively a flat ground. The mean value is taken out of the mean differences from the single joints.

accurate on average. Furthermore, there is a large difference between the simulated torque of the moving joints represented by a 1.2 Nm decrease. The other evaluated parameters also decreased in their accuracy. So the execution of the animation on the real robot on grass does not decrease the simulation accuracy. That could have been an influence because of the comparison with the on a flat ground executed simulated motion execution. One explanation for this is the vibrating of the robot in reality, which is higher on flat ground than on damping grass. Because the simulation is not realistically modeling this effect, the simulation is closer to the reality when the robot is standing on grass.

Other influences are vibrations due to the robots motion, collision forces due to the contact with the ground and influences due to the load acting on the robot. Whether this is simulated correctly depends on the physics engine. In this thesis, ODE [O22] was used, which does not support the robots kinematic of links and joints. Joints are modeled as constraints between bodies, in this case, the links [40]. Because of this,

the robot's movements have to be calculated in the Cartesian space, and calculation in the joint space is not possible. On the other hand ODE is strong in modeling collisions [40]. Still, ODE or any other existing physics engine is not modeling the physics acting in reality exactly correct [40]. Therefore with improving the robot description file, it is not possible to achieve a simulation, which bridges the Reality Gap completely. Even if the robot description file is stating the robots features correct, the simulation would not be accurate, because of the inaccuracies of the physics engine.

Possibly another physics engine would simulate the regarded robot more accurately. The used simulator Gazebo supports, besides ODE, the physics engines Bullet, Simbody, and DART. However, only ODE is fully integrated into the simulator. Furthermore, the with ODE running setup of the simulator is not working combined with other physics engines. The usage of Bullet instead of ODE caused the robot to hover slightly over the ground and moving with a small impulse in one direction. Thereby the robot moved its legs, without a command to do so. That resulted in a by the simulation not noticed self-collision of the feet and lower legs. When using ODE, the robot is standing in this situation. Starting the simulator with Simbody lead to a real-time factor of 0.0, the clock in simulation did not run, and the world setup was completed with mistakes, e.g., the goal net over the center circle. The physics engine DART requires Gazebo to be installed from source. After the simulation is started with DART, the robot is shortly standing, but it starts to crock. That generates an impulse leading to the robot rotating about the front edge of its feet. Therefore the robot is falling. Presumably, this impulse is occurring because of resulting collision forces due to the robot sinking a little in the ground. Hence, at the current state, the only easily usable physics engine is ODE.

The limited computational power is also a reason for an inaccurate simulation as this is limiting the possible update rate. A real-time factor, which symbolizes the running speed of the simulation in comparison to the time passing in reality, should be not too small, so a motion execution does not take a lot longer than in reality. For achieving that, even if the simulation is slower than reality, the solver iterations and update rate of the physics engine have to be limited. So a trade-off between accuracy and usability has to be used. That leads to a not continuous simulation, which is only updated at specific time steps to mimic the real world. In between the reality differs a little from the simulated state causing inaccuracies in the simulation.

Furthermore, the reality is not wholly deterministic. For example, the motors are influenced by the backlash phenomena, which leads to the fact that the motors do not reach exactly the position stated in the control software. Also, the behavior of each motor is not entirely the same. In a description file, only one value describes a characteristic, e.g., the motor's torque. The in the robot description file stated parameters are determined out of the values from some test motors which already differed in their strength. With using more data to find the parameters for simulation describing the motor, they could gain accuracy. However, overall, this would only lead to a better average simulation, while the single motors included in the robot can still differ from these values. Finding out the correct parameters for every motor alone would be an option, but this is

too much of a workload as sometimes motors break and get changed and the motors are affected by wear which continually changes the parameters a little. Moreover, errors occurring during a measurement have a higher impact on the results when the motors are regarded individually.

A further impact on the motors is the input voltage, which influences the torque and velocity of the motors. With just one parameter each in the simulation, it is not possible to model the different behavior of the robot under the influence of different input voltages. Therefore, using the robot with an accumulator will always lead to changes between simulation and reality. The simulation could be adapted to the voltage provided by the external brick of the robots, but as this is not used during a soccer game, it is, in case of the Wolfgang robot, not too valuable. Still, this can lead to differences between simulation and reality. Because of the voltage drop occurring in the robots chains, the maximum torque and velocity also change per motor. When a function to describe the voltage drop would be known, this could be regarded in the affected parameters of the robot description file. At the current state, this leads to a difference between simulation and reality.

Overall it is not possible to state with certainty that the in this thesis found parameters are describing the robot model accurately. For example, the mass and inertia tensor of a link could be stated incorrectly, but due to an incorrect motion simulation, the parameters seem correct. Still, for the current state, an improvement was made to simulate the robot's motion. It is possible to use the simulator to check if a motion sequence is correctly implemented or to check if different software components work integrated. However, the robot is more stable in simulation, and the joints behavior is not yet modeled as in reality. Therefore the real robot is still necessary to evaluate if a generated motion sequence can be executed on the robot without the robot falling. A further disadvantage is the speed of the simulation as it runs significantly slower than in reality.

7. Conclusion

In this thesis, it was decided to improve the robot description file of the Wolfgang robot platform as this seemed to be a valuable first step, which was done by other researchers as well to minimize the Reality Gap. The physical parameters of the robot's links and the physical parameters for the joints, except for the dynamic friction, have been updated. For this updates, the links of the robots and the robot's motors, representing its joints, were studied. In Section 7.1, the results are summarized, and a conclusion is drawn. Further work needed to close the Reality Gap further is presented in Section 7.2.

7.1. Summary

The improvements of the chosen physical parameters in the robot description file of the Wolfgang robot platform lead to an overall enhancement of the simulation accuracy. In general, the robot is behaving more realistic in simulation regarding the motion execution because of the improvements. The improvements consist of an examination of the robot's links regarding their mass, their center of mass, and their inertia tensor. Besides the into the joints integrated motors were analyzed. Due to this analysis, the torque, angular velocity, and static friction are now described more accurately in the robot description file. As a result of these improvements, the joint's position and torque values are modeled more realistic. Just the joints velocity simulation decreased due to influences like the PID control algorithm.

Additional investigations figured out that the motor's behavior is impacted by lots of different aspects. For example, the input voltage profoundly affects the for a motor maximal achievable torque and angular velocity. That impacts the robot due to a voltage drop in the robots chains. This voltage drop increases due to high load on the robot. In general, the load already has an effect on a motor's performance.

Overall, the simulation of the Wolfgang robot platform is better usable now, which symbolizes a minimization of the Reality Gap. For ultimately bridging the Reality Gap, further work is needed. The decision to improve the robot description file and the methods used to find the physical parameters were useful as they lead to an improvement of the simulation. Because the used methods are generally applicable, they can be used for other robots as well to improve the physical characteristics in their robot description file and thereby achieve a more accurate simulation of the regarded robot.

7.2. Future Work

For keeping the level of simulation accuracy, it is necessary to update the parameters in the robot description file when changes are made to the robot. So for example, when the robot gets a new head link for a new camera or when a new computation unit is added to the torso of the robot, the physical characteristics of the impacted links change. So the parameters describing the impacted links have to be adapted to the new structure and weight.

For bridging the Reality Gap further, there are lots of approaches, which can be pursued. In general, some are explained in chapter 2. Regarding the Wolfgang robot, it would be interesting to analyze the forces and vibrations within the robot as well, as the forces occurring because of collisions. That could be used to determine the load acting on a motor of the robot more precisely. Moreover, creating a model of the grass for simulation could be helpful. Simulating the collision forces between the feet and grass should result in a more accurate simulation of the robot's motion. It would also be interesting to investigate if the other physics engines are simulating these effects and the robot, in general, more accurately. For realization, a working simulation setup has to be found, and possibly some crucial functionalities have to be integrated into Gazebo. A significant step is to determine useful values for the update rate and solver iterations. That should also be investigated further for the used physics engine ODE. Moreover, the force transfer from the motor horn to the link depends on how tight the screws are. This impact on the robot's behavior should also be analyzed.

Furthermore, a study of the input voltage of the motors would be interesting as it has been figured out in this thesis that there is a voltage drop in the robot. Moreover, the input voltage has an effect on the torque and velocity the motors can produce. Therefore a further analysis would be helpful to create a more accurate actuator model.

This actuator model can be written as a plugin for Gazebo and then be used to include, for example, the influences of the input voltage and the load or other impacts to the actuator in the simulation. Also, motor characteristics, e.g., backlash, resolution, gear ratio, or internal friction in the motor could be described. The implementation of the PID controller should also be included in this model because this impacts the motor control significantly. For this, it has to be investigated further how the PID controller of the motors is implemented. Besides a deeper knowledge of the influence of the P, I and D values on the in simulation used controller should be gained, to be able to use the robot's PID values in simulation. A useful guideline are the joint position controllers for simulation in the *ros_controller* package [O12] because the at the current state used PID controller is implemented in this package. Furthermore, for a simulation of a real game, it would be nice to be able to simulate the accumulator operation as the voltage drop impacts what the robot can still do. An example of this is the stand up of the robot from the ground. Successful execution can get hard for the robot near the end of halftime.

A further effect, which would be interesting to study and then also include in the actuator model is the impact of the temperature of the motors on the robot's behavior. Possibly there is a difference between a motor at room temperature and a motor which heated due to usage. That the motors get warm also happens during a game and when, e.g., overload errors occur and can, therefore, impact the robot's behavior.

All in all, there are still lots of physical aspects unknown or unclear, which can be studied and then implemented. That will help to improve the simulation accuracy and therefore to bridge the Reality Gap. Besides, a deeper understanding of the robot is gained.

Bibliography

[1] Abbas Abdolmaleki et al. “Learning a Humanoid Kick with Controlled Distance”. In: *RoboCup 2016: Robot World Cup XX*. Ed. by Sven Behnke et al. Cham: Springer International Publishing, 2017, pp. 45–57. ISBN: 978-3-319-68792-6.

[2] Dorn Bader. *Physik. Gymnasium Gesamtband Sek 2*. Schroedel, 2009, pp. 246–247.

[3] Constantin Bergatt. “Minimierung und Untersuchung des Simulation-Reality-Gaps anhand eines BRIOR©-Labyrinth-Spiels”. Master Arbeit. Universität Rostock Fakultät für Informatik und Elektrotechnik Institut für Mikroelektronik und Datentechnik in Zusammenarbeit mit dem Deutsches Forschungszentrum für Künstliche Intelligenz GmbH Forschungsbereich Robotik Bremen, März 2009.

[4] Andreas Bihlmaier and Heinz Wörn. “Hands-on Learning of ROS Using Common Hardware”. In: *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Ed. by Anis Koubaa. Cham: Springer International Publishing, 2016, pp. 44–46. ISBN: 978-3-319-26054-9. DOI: 10.1007/978-3-319-26054-9_2. URL: https://doi.org/10.1007/978-3-319-26054-9%5C_2.

[5] A. C. Bittencourt et al. “An extended friction model to capture load and temperature effects in robot joints”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2010, pp. 6161–6167. DOI: 10.1109/IROS.2010.5650358.

[6] Yevgen Chebotar et al. “Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience”. In: *CoRR* abs1810.05687 (2018). arXiv: 1810.05687. URL: http://arxiv.org/abs/1810.05687.

[7] J. Chen. “The effects of gear reduction on robot dynamics”. In: *JPL, California Inst. of Tech., Proceedings of the NASA Conference on Space Telerobotics*. Vol. 4. 1989, pp. 297–307.

[8] Sachin Chitta et al. “ros_control: A generic and simple control framework for ROS”. In: *The Journal of Open Source Software* 2.20 (Dec. 2017), pp. 456–456. DOI: 10.21105/joss.00456. URL: https://hal.archives-ouvertes.fr/hal-01662418.

[9] Siebe Claes. “Transferring learned control from simulation to real compliant quadrupeds for robust locomotion using CPG controllers”. Master’s dissertation. Ghent University, Department of Electronics, Information Systems, Faculty of Engineering, and Architecture, 2017.

[10] Paulo Costa et al. “SimTwo Realistic Simulator: A Tool for the Development and Validation of Robot Software”. In: *Theory and Applications of Mathematics & Computer Science* 1 (Apr. 2011), pp. 17–33.

[11] Wolfgang Demtröder. *Experimentalphysik 2. Elektrizität und Optik*. Vol. 5. Springer, 2009, p. 59.

[12] Wolfgang Demtröder. *Experimentalphysik 2. Elektrizität und Optik*. Vol. 5. Springer, 2009, pp. 145–147.

[13] Alexander Denisov, Victor Budkov, and Daniil Mikhalchenko. “Designing Simulation Model of Humanoid Robot to Study Servo Control System”. In: *Interactive Collaborative Robotics*. Ed. by Andrey Ronzhin, Gerhard Rigoll, and Roman Meshcheryakov. Cham: Springer International Publishing, 2016, pp. 69–78. ISBN: 978-3-319-43955-6.

[14] Evan Drumwright et al. “Extending Open Dynamics Engine for Robotics Simulation”. In: *Simulation, Modeling, and Programming for Autonomous Robots*. Ed. by Noriaki Ando et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 38–50. ISBN: 978-3-642-17319-6.

[15] Amir Gholami, Milad Moradi, and Majid Majidi. “A simulation platform design and kinematics analysis of MRL-HSL humanoid robot”. In: *RoboCup 2019: Robot World Cup XXIII*. To appear. Springer, 2019.

[16] Douglas C. Giancoli. *Physik. Lehr- und Übungsbuch*. Vol. 3. PEARSON, 2010, pp. 903–904.

[17] Florian Golemo et al. “Sim-to-Real Transfer with Neural-Augmented Robot Simulation”. In: *Proceedings of The 2nd Conference on Robot Learning*. Ed. by Aude Billard et al. Vol. 87. Proceedings of Machine Learning Research. PMLR, 29–31 Oct 2018, pp. 817–828. URL: http://proceedings.mlr.press/v87/golemo18a.html.

[18] Serena Ivaldi, Vincent Padois, and Francesco Nori. “Tools for dynamics simulation of robots: a survey based on user feedback”. In: *CoRR* abs/1402.7050 (2014). arXiv: 1402.7050. URL: http://arxiv.org/abs/1402.7050.

[19] Shuuji Kajita et al. “ZMP and Dynamics”. In: *Introduction to Humanoid Robotics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 69–103. ISBN: 978-3-642-54536-8. DOI: 10.1007/978-3-642-54536-8_3. URL: https://doi.org/10.1007/978-3-642-54536-8%5C_3.

[20] N. Koenig and A. Howard. “Design and use paradigms for Gazebo, an open-source multi-robot simulator”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. Sept. 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

[21] Matthias Kubisch. “Modellierung und Simulation nichtlinearer Motoreigenschaften”. Studienarbeit. Humboldt-Universität zu Berlin, Institut für Informatik, Lehrstuhl für künstliche Intelligenz, 16. Juni 2008.

[22] Tim Laue and Matthias Hebbel. “Automatic Parameter Optimization for a Dynamic Robot Simulation”. In: *RoboCup 2008: Robot Soccer World Cup XII*. Ed. by Luca Iocchi et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 121–132. ISBN: 978-3-642-02921-9.

[23] José L. Lima et al. “Humanoid Gait Optimization Resorting to an Improved Simulation Model”. In: *International Journal of Advanced Robotic Systems* 10.1 (2013), p. 67. DOI: 10.5772/54766. eprint: https://doi.org/10.5772/54766. URL: https://doi.org/10.5772/54766.

[24] José Lima et al. “Humanoid realistic simulator: the servomotor joint modeling”. In: *In 6th International Conference on Informatics in Control, Automation and Robotics. Milan.* 2009.

[25] Jörn Malzahn, Victor Barasuol, and Klaus Janschek. “Actuator Modeling and Simulation”. In: *Humanoid Robotics: A Reference*. Ed. by Ambarish Goswami and Prahlad Vadakkepat. Dordrecht: Springer Netherlands, 2018, pp. 1–48. ISBN: 978-94-007-7194-9. DOI: 10.1007/978-94-007-7194-9_75-1. URL: https://doi.org/10.1007/978-94-007-7194-9%5C_75-1.

[26] Norbert Michael Mayer et al. “3D2Real: Simulation League Finals in Real Robots”. In: *RoboCup 2006: Robot Soccer World Cup X*. Ed. by Gerhard Lakemeyer et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 25–34. ISBN: 978-3-540-74024-7.

[27] J. G. McAnanama and G. Marsden. “An open source flight dynamics model and IMU signal simulator”. In: *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. Apr. 2018, pp. 874–881. DOI: 10.1109/PLANS.2018.8373465.

[28] Arno Mensink. “Characterization and modeling of a Dynamixel servo”. Individuel Research Assignment. University of Twente Faculty Control Engineering, November 2008.

[29] Johannes Meyer et al. “Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo”. In: *Simulation, Modeling, and Programming for Autonomous Robots*. Ed. by Itsuki Noda et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 400–411. ISBN: 978-3-642-34327-8.

[30] Minh Thao Nguyenová, Petr Čížek, and Jan Faigl. “Modeling Proprioceptive Sensing for Locomotion Control of Hexapod Walking Robot in Robotic Simulator”. In: *Modelling and Simulation for Autonomous Systems*. Ed. by Jan Mazal. Cham: Springer International Publishing, 2019, pp. 215–225. ISBN: 978-3-030-14984-0.

[31] Xue Bin Peng et al. “DeepMimic: Example-guided Deep Reinforcement Learning of Physics-based Character Skills”. In: *ACM Trans. Graph.* 37.4 (July 2018), 143:1–143:14. ISSN: 0730-0301. DOI: 10.1145/3197517.3201311. URL: http://doi.acm.org/10.1145/3197517.3201311.

[32] C. J. Pretorius, M. C. du Plessis, and J. W. Gonsalves. “The transferability of evolved hexapod locomotion controllers from simulation to real hardware”. In: *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. July 2017, pp. 567–574. DOI: 10.1109/RCAR.2017.8311923.

[33] W. Qian et al. “Manipulation task simulation using ROS and Gazebo”. In: *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*. Dec. 2014, pp. 2594–2598. DOI: 10.1109/ROBIO.2014.7090732.

[34] M. Quigley et al. “" ROS: an open-source robot operating system"”. In: *ICRA Workshop on Open Source Software*. 2009.

[35] Max Schwarz et al. “Humanoid TeenSize Open Platform NimbRo-OP”. In: *RoboCup 2013: Robot World Cup XVII*. Ed. by Sven Behnke et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 568–575. ISBN: 978-3-662-44468-9.

[36] Lorenzo Sciavicco, Bruno Siciliano, and Villani Luigi. “Lagrange and Newton-Euler dynamic modeling of a gear-driven robot manipulator with inclusion of motor inertia effects”. In: *Advanced Robotics* 10 (Jan. 1995), pp. 317–334. DOI: 10.1163/156855395X00427.

[37] Eckhard Spring. “Gleichstrommaschine”. In: *Elektrische Maschinen: Eine Einführung*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–98. ISBN: 978-3-642-00885-6. DOI: 10.1007/978-3-642-00885-6_1. URL: https://doi.org/10.1007/978-3-642-00885-6%5C_1.

[38] J. Tan et al. “Simulation-based design of dynamic controllers for humanoid balancing”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2016, pp. 2729–2736. DOI: 10.1109/IROS.2016.7759424.

[39] Jie Tan et al. “Sim-to-Real: Learning Agile Locomotion For Quadruped Robots”. In: *CoRR* abs/1804.10332 (2018). arXiv: 1804.10332. URL: http://arxiv.org/abs/1804.10332.

[40] Barkan Ugurlu and Serena Ivaldi. “Free Simulation Software and Library”. In: *Humanoid Robotics: A Reference*. Ed. by Ambarish Goswami and Prahlad Vadakkepat. Dordrecht: Springer Netherlands, 2018, pp. 1–20. ISBN: 978-94-007-7194-9. DOI: 10.1007/978-94-007-7194-9_27-1. URL: https://doi.org/10.1007/978-94-007-7194-9%5C_27-1.

[41] Erik Weitnauer, Robert Haschke, and Helge Ritter. “Evaluating a Physics Engine as an Ingredient for Physical Reasoning”. In: *Simulation, Modeling, and Programming for Autonomous Robots*. Ed. by Noriaki Ando et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 144–155. ISBN: 978-3-642-17319-6.

[42] Yuan Xu, Heinrich Mellmann, and Hans-Dieter Burkhard. “An Approach to Close the Gap between Simulation and Real Robots”. In: *Simulation, Modeling, and Programming for Autonomous Robots*. Ed. by Noriaki Ando et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 533–544. ISBN: 978-3-642-17319-6.

[43] H. Yang et al. “Dynamics verification of free-floating space robot based on the hybrid simulation”. In: *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*. Dec. 2014, pp. 2327–2332. DOI: 10.1109/ROBIO.2014.7090685.

[44] Xinglong Yang et al. “An Efficient Simulation Platform for Testing and Validating Autonomous Navigation Algorithms for Multi-rotor UAVs Based on Unreal Engine”. In: *China Satellite Navigation Conference (CSNC) 2019 Proceedings*. Ed. by Jiadong Sun, Changfeng Yang, and Yuanxi Yang. Singapore: Springer Singapore, 2019, pp. 527–539.

[45] W. Yao et al. “A simulation system based on ROS and Gazebo for RoboCup Middle Size League”. In: *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. Dec. 2015, pp. 54–59. DOI: 10.1109/ROBIO.2015.7414623.

[46] Brandon Zahn et al. “Optimization of robot movements using genetic algorithms and simulation”. In: *RoboCup 2019: Robot World Cup XXIII*. To appear. Springer, 2019.

[47] J. Zhang et al. “Virtual model optimization and locomotion control of bionic hexapod robot”. In: *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. July 2012, pp. 497–501. DOI: 10.1109/ICIEA.2012.6360779.

Online References

[O1] *Autodesk Inventor*. Accessed: 12.06.19. URL: https://www.autodesk.de/products/inventor/overview.

[O2] *Breve Simulation Environment*. Accessed: 10.06.2019. URL: http://www.spiderland.org/s/.

[O3] *Bullet Real-Time Physics Simulation*. Accessed: 06.05.2019. URL: https://pybullet.org/wordpress/.

[O4] *DXL Board specifications for motor communication*. Accessed: 15.06.2019. URL: https://github.com/Rhoban/DXLBoard.

[O5] *Dynamic Animation and Robotics Toolkit*. Accessed: 06.05.2019. URL: http://dartsim.github.io/.

[O6] *Dynamixel Control Protocol*. Accessed: 15.06.2019. URL: http://emanual.robotis.com/docs/en/dxl/protocol2/.

[O7] *Explanation of Autodesk Inventors physical properties calculation (Message 3)*. Accessed: 12.06.19. URL: https://forums.autodesk.com/t5/inventor-forum/mass-moment-of-inertia/td-p/3532526.

[O8] *Explanation of moment of inertia calculation*. Accessed: 20.07.2019. URL: http://mathworld.wolfram.com/MomentofInertia.html.

[O9] *Explanation of Rheological Models*. Accessed: 18.06.2019. URL: http://homepages.engineering.auckland.ac.nz/~pkel015/SolidMechanicsBooks/Part_I/BookSM_Part_I/10_Viscoelasticity/10_Viscoelasticity_03_Rheological.pdf.

[O10] *Figure of Wolfgang robot*. Accessed: 19.06.2019. URL: http://doku.bit-bots.de/meta/_images/motors.png.

[O11] *Github Hamburg Bit-Bots*. Accessed: 13.06.2019. URL: https://github.com/bit-bots.

[O12] *Implementation of ROS controllers*. Accessed: 21.07.2019. URL: https://github.com/ros-controls/ros_controllers.

[O13] *Implementation of the in simulation used motor controller*. Accessed: 17.07.2019. URL: https://github.com/ros-controls/ros_controllers/blob/melodic-devel/effort_controllers/src/joint_group_position_controller.cpp.

[O14] *Introduction to ROS*. Accessed: 09.05.2019. URL: http://wiki.ros.org/ROS/Introduction.

[O15] *List of Tools used in RoboCup 3D Simulation League*. Accessed: 11.06.2019. URL: https://ssim.robocup.org/3d-tools/.

[O16] *Manual of Dynamixel MX 106*. Accessed: 15.06.2019. URL: http://emanual.robotis.com/docs/en/dxl/mx/mx-106-2/#moving-threshold.

[O17] *Manual of Dynamixel MX 64*. Accessed: 15.06.2019. URL: http://emanual.robotis.com/docs/en/dxl/mx/mx-64-2/#moving-threshold.

[O18] *MARS Simulator*. Accessed: 08.06.2019. URL: https://robotik.dfki-bremen.de/en/research/softwaretools/mars.html.

[O19] *MIT course: DC motor characteristics*. Accessed: 16.06.2019. URL: http://lancet.mit.edu/motors/motors3.html.

[O20] *MIT course: inertia tensor for ridgid body dynamics*. Accessed: 15.06.2019. URL: https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-07-dynamics-fall-2009/lecture-notes/MIT16_07F09_Lec26.pdf.

[O21] *MIT course: servo motors with backlash*. Accessed: 15.06.2019. URL: https://ocw.mit.edu/courses/mechanical-engineering/2-017j-design-of-electromechanical-robotic-systems-fall-2009/assignments/MIT2_017JF09_p32.pdf.

[O22] *Open Dynamics Engine*. Accessed: 06.05.2019. URL: http://ode.org/.

[O23] *Open Source Robotics Foundation*. Accessed: 06.05.2019. URL: https://www.openrobotics.org/.

[O24] *Picture: ros_control control loop in Gazebo and reality*. Accessed: 07.05.2019. URL: https://bitbucket.org/osrf/gazebo_tutorials/raw/default/ros_control/Gazebo_ros_transmission.png.

[O25] *Roadmap of RobCup humanoid soccer*. Accessed: 13.12.2018. URL: https://www.robocuphumanoid.org/wp-content/uploads/HumanoidLeagueProposedRoadmap.pdf.

[O26] *RoboCup webpage*. Accessed: 18.06.2019. URL: https://www.robocup.org/.

[O27] *Robot Operating System*. Accessed: 08.05.2019. URL: http://www.ros.org/.

[O28] *Robotis Company*. Accessed: 15.06.2019. URL: http://www.robotis.us/.

[O29] *ROS interface for ATI F/T sensors*. Accessed: 15.06.2019. URL: https://github.com/UTNuclearRoboticsPublic/netft_utils/tree/master/include.

[O30] *ROS Wrapper for Gazebo*. Accessed: 06.05.2019. URL: http://wiki.ros.org/gazebo_ros_pkgs.

[O31] *ROSdyn project*. Accessed: 10.06.2019. URL: http://rosin-project.eu/ftp/rosdyn.

[O32] *Simbody: Multibody Physics API*. Accessed: 06.05.2019. URL: https://simtk.org/projects/simbody.

[O33] *Simulation Description Format*. Accessed: 06.05.2019. URL: http://sdformat.org/.

[O34] *Simulation Software Gazebo*. Accessed: 28.04.2019. URL: http://gazebosim.org/.

[O35] *Simulator V-REP*. Accessed: 09.06.2019. URL: http://coppeliarobotics.com/.

[O36] *Simulink Software*. Accessed: 09.06.2019. URL: https://de.mathworks.com/products/simulink.html.

[O37] *Specifications of ATI F/T Sensor Mini 45*. Accessed: 15.06.2019. URL: https://www.ati-ia.com/products/ft/ft_models.aspx?id=Mini45.

[O38] *Technical Specification of the Wolfgang robot platform*. Accessed: 14.06.2019. URL: https://submission.robocuphumanoid.org/uploads//Hamburg_Bit_Bots_and_WF_Wolves-specs-5c03d58ec8f93.pdf.

[O39] *Visualization of the position controller of the Dynamixel MX series motors by Robotis*. Accessed: 21.07.2019. URL: http://emanual.robotis.com/assets/images/dxl/position_controller_pid_gain.jpg.

A. Appendix

A.1. Center of Mass Measurements

link	measured			Autodesk		
	x position	y position	z position	x position	y position	z position
torso	0.007	-0.002	0.021	0.014563	0.001600	0.016559
hip_1	0.000	0.000	-0.011	-0.000001	0.000000	-0.014563
hip_2	0.046	-0.002	-0.026	0.051391	0.000232	-0.014250
upper_leg	-0.006	0.004	-0.126	-0.004721	-0.000424	-0.132759
lower_leg	0.012	0.000	-0.076	0.005873	-0.000102	-0.093305
ankle	-0.020	-0.005	0.026	-0.017874	0.000190	0.013809
foot	0.081	-0.016	-0.033	0.075576	-0.013868	-0.035997
shoulder	-0.012	0.020	0.000	-0.010164	0.022268	-0.000376
upper_arm	0.000	0.000	0.031	-0.000158	0.000012	0.035495
lower_arm	0.055	0.000	0.001	0.050639	-0.000164	0.000492
neck	-0.005	0.000	0.016	-0.027758	-0.000252	0.032356
head	0.004	0.000	0.096	0.012794	-0.000380	0.100180

Table A.1.: A look-up table for the center of mass of the Wolfgang robots links is shown. On the left side in the measured column, the experimentally determined x, y and z positions can be found, and on the right side, the values outputted from Autodesk can be found. All values are stated in the unit meter [m].

A.2. Mass Values of the Wolfgang Robots' Links

link	stated mass	measured mass	relative error
torso	1.8051	2.889	0.600465348180156
hip_1	0.073374	0.098	0.335622972715131
hip_2	0.35844	0.349	-0.026336346389912
upper_leg	0.38276	0.248	-0.352074406939074
lower_leg	0.20549	0.085	-0.586354567132221
ankle	0.35903	0.349	-0.027936384146172
foot	0.1488	0.312	1.09677419354839
shoulder	0.025434	0.097	2.81379256113863
upper_arm	0.2575	0.229	-0.110679611650485
lower_arm	0.23502	0.208	-0.114968938813718
neck	0.15587	0.153	-0.01841277988067
head	0.40746	0.110	-0.73003485004663

Table A.2.: A look-up table for comparing the in the given URDF stated mass with the new weighed values is shown. The mass values are stated in the unit [kg].

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Bachelorstudiengang Computing in Science selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel — insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen — benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Hamburg, den 24.07.2019 Tanja Flemming

Veröffentlichung

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

Hamburg, den 24.07.2019 Tanja Flemming